POSE ESTIMATION FOR FUTURE PREDICTION OF FALLING

Evan Kurpiewski

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Congdon School of Supply Chain, Business Analytics, and Information Systems

University of North Carolina Wilmington

2022

Approved by

Advisory Committee

Karl Ricanek	Minoo Modaresnezhad
Yang Song	Gulustan Dogan Chair

Accepted by

Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

Background: In this paper, I propose the use of various deep learning and artificial intelligence techniques to evaluate the detection and possible prediction of an individual falling taken from images or video. Falls are a serious issue within the medical sector [1]. While falls are often associated with senior centers and retirement homes, falls are responsible for approximately 37.3 million injuries severe enough for medical attention [2]. The goal of this thesis is to utilize human pose estimation and its output into another network or networks in an attempt to detect and predict falls that may occur. In addition, there is a number of additional possibilities such as utilizing angles between the joints or attempting to predict the next sequence of key points as an indicator of chance to fall which may be useful in an effort to catch individuals before they fall and thus ameliorate the risk to health that falls cause.

Objectives: The primary purpose of this thesis is utilizing various techniques and deep learning models to determine if falls can be predicted or determined before they occur. To develop such a process, a literature review will be undertaken to understand the best possible methodology to achieve this objective. In addition, the creation of a new dataset with joint key points will be created and also tested solely for the purpose of determining the efficacy of detecting falls purely from key points.

Methods: This thesis utilizes a fall detection dataset whereby 70 videos of subjects falling and not falling were recorded. These videos have been hand labeled on a pure frame basis as to whether the subject was falling or not falling. By applying a general-purpose human pose estimation system to these videos, we can gather key points of joint locations. From these various calculations, models can be made to achieve the objectives of this thesis to detect and predict falls.

Results: At the end of this thesis, it was determined that human pose estimation has gathered enough general capacity to accurately create key points for videos and images. In order to evaluate performance, F1 score is utilized. F1 is the currently accepted best single measure of performance of a classification model with a score of 1 being perfect. From these aforementioned key points, falls could be accurately detected with a high F1 score of .88. In addition, falls could be predicted one frame ahead of time at a decent performance, consisting of an F1 score of .80.

Conclusions: From the results gathered, it was concluded that falls can be detected from human pose estimation and that human pose estimation may provide additional benefit from simply evaluating the image as a whole. In addition, it was concluded that falls could be predicted ahead of time, however, more exploration is needed in this area in order to determine how far in the future this prediction could be made and how accurate the prediction could become.

ACKNOWLEDGMENTS

I would like to thank Dr.Gulustan Dogan, my advisor, for introducing me to this area of study and for guiding me in my research. I would like to thank my committee members, Dr.Ricanek, Dr.Song, and Dr.Modaresnezhad for their intellectual contributions to my thesis.

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges-2 system, which is supported by NSF award number ACI-1928147, at the Pittsburgh Supercomputing Center (PSC).

DEDICATION

I would like to dedicate this thesis to my wife who stood by me throughout this academic journey. Without her aid and gentle guidance throughout this process, I would have found this endeavor impossibly difficult.

LIST OF TABLES

LIST OF FIGURES

Introduction

Motivation

Falls are one of the most consistent threats to human health, especially among the elderly and are a major public health problem, where 1 in 4 elderly individuals fall each year [1]. To make matters worse, 1 in 5 of these falls lead to a serious injury such as a broken bone or a traumatic head injury [1]. In addition, while falls are often scoped around the elderly, there are approximately 37.3 million injuries per year due to falls when younger individuals are included [2] [3] [4].

As noted, another group that is at risk for health consequences due to falls is children [1]. While children have many more falls that occur due to evolving developmental stages and innate curiosity, they are often at high risk, as falls can lead to head trauma or broken bones that lead to health issues later in life. While children are often better at recovering from falls in comparison to the elderly, preventing falls in all individuals would reduce health risks and consequences while also mitigating costs in our healthcare system.

In both of these groups, there is a need to ameliorate the potential damage and health consequences of falls. Fortunately, in recent years, there has been a boon of fall detection devices. Devices such as life alert that allow elderly individuals to get help quickly can mitigate the loss of life or major injury due to falls whereby a long lie occurs. A long lie in this context means that a person has fallen and is unable to quickly receive attention which often increases the consequences of the fall. Another common device for detecting falls is the Apple Watch. However, these devices have a common drawback in that they rely on being worn and utilizing accelerometers and gyrometers to determine a fall.

Another method of determining falls is through video footage. This field is a bit more nascent and while there is some research that has occurred it is often scoped within specific niche circumstances such as a depth camera that does not work similarly to regular RGB cameras most people are familiar with. Due to these niche circumstances, the fall detection methods that currently exist do not provide strong, general capability for fall detection.

In addition, the above-mentioned methods only detect falls which will allow for quick response to the fall. While this is beneficial, the ideal system would predict or estimate a fall before it has occurred. A system that could determine whether an individual has a high chance of falling in the near future from general-purpose, RGB camera footage would provide a powerful and consistent method to prevent health risks from falls for individuals of all ages.

Both wearable and visual fall detection fields rely heavily on the use of machine learning and deep learning. These techniques can analyze large volumes of data to determine the appropriate function within which to determine whether an individual has fallen. In addition, various datasets are utilized, however, it should be noted that most datasets at this conjuncture utilize simulated falls which may affect general-purpose capacity of these techniques [5].

In order to properly evaluate falling, the pose of the individual should be evaluated for potential risks such as those that may be present in their gait. Human pose estimation is a technique that would better enable the evaluation of the pose for such problems. The field of human pose estimation has been booming in recent years. Within this field, machine learning and deep learning algorithms are able to determine where a human is within an image or sequence of image. From this they can create key points for the joint locations of the individual. These joint locations can be utilized to derive other important variables and features such as angles of the joints themselves while also reducing the complexity of the problem by eliminating the need to evaluate the entire image from pixel information.

Aim

The aim of this thesis is to predict whether a person has the possibility of falling within the near future from visual information. The prediction is performed using computer vision techniques to detect their pose which can then be used to deduce angles and future positioning of pose key points in an attempt to predict and detect falls.

Objectives

The main objectives of this thesis are,

- Enhancing an existing fall detection dataset such that contains pose estimation key points for joints along with angles for improved performance.
- Creating a deep learning neural network that can detect/predict falls from key points output from a pose estimation network.
- The derivation of angles from key points output from a pose estimation network to improve fall detection and prediction performance.
- A time series forecasting network capable of predicting the next row of a dataset from previous row information.
- The combination of the above networks to provide a potential view into the feasibility of fall prediction.

Research Questions

To achieve the objectives of our thesis, some research questions have been formulated:

1. Can the output of a pose estimation network be used to detect falls?

   Motivation: Pose estimation offers some potential improvements over fall detection from pure pixel information. If fall detection can be performed via pose estimation, these improvements may be realized for fall prediction as well. [6] [7] [8] [9].

2. Can the calculation of angles from key points of a pose estimation network aid in the detection and prediction of falls?

   Motivation: The ultimate goal of is to be able to predict falls before they occur. When evaluating this problem, it became apparent that key points alone may not provide enough information for accurate detection and prediction. However, angles derived from the key

points may provide additional context for the deep neural network to determine fall potential.

3. Can the output of a pose estimation network be time-series forecast to predict the next row or rows that would be output?

   Motivation: In order to predict falls, we need to estimate into the future. This would require the prediction of where key points and angles would be next. This would utilize time series forecasting to predict and estimate the next row within the series of key points and angles.

4. Can the combination of time series forecasting and fall detection on key points from pose estimation provide future prediction of falling?

   Motivation: This is the combination of the above questions in an attempt to answer the ultimate goal of this research – can we predict that a fall is going to occur before it actually occurs.

Scope

This research focuses on the development of models for fall detection and prediction. It does not focus on creating brand new pose estimation networks, though a strong understanding is needed there as well. In addition, there is no expectation of performance of models in terms of run-time though this is an aspect being considered. Lastly, the goal is to predict into the future to show its possibility, however, there is no expectation of time length into the future within this thesis. Instead, the goal is the provide proof that prediction is feasible.

Organization

This thesis is organized into different chapters which are as follows:

- Chapter 1: This chapter contains the introduction to this thesis, aim, objectives, research questions, and underlying motivations that have driven this research.

- Chapter 2: In this chapter, the background of the concepts used during the research is discussed.
- Chapter 3: This chapter contains related works and a literature review to the thesis at hand which act as a guiding light and inspiration for the research conducted. It also highlights lessons learned from these pieces of research which we intend to utilize to improve upon existing research.
- Chapter 4: In this chapter, the methods used to answer the research questions are discussed. It includes experimental analysis such as data processing that took place between and before steps within the project. In addition, it details the software setup that was necessary to perform this work as well as the data used itself.
- Chapter 5: The results of the research are presented within this chapter.
- Chapter 6: This chapter contains an analysis and discussion of the results and methods used as well as the contribution to existing research.
- Chapter 7: The final chapter discusses the conclusion of the thesis as well as discussion on future work and where this research can lead us as a community.

Background

In recent years, there has been a large introduction and push towards artificial intelligence (AI), machine learning, and deep learning. Machine learning and deep learning are both subsets of the larger AI field. These techniques developed from pattern recognition techniques, often mimicking processes of the human brain such as in artificial neural networks. The field of AI has become large and expansive, growing rapidly in recent years. This has been enabled by the advent of "big data" [10]. Big Data is the serendipitous collection of data at large by many organizations worldwide [10]. Subsequently, techniques such as machine learning and deep learning, which can analyze and utilize these large amounts of data, have blossomed into powerful tools for discerning patterns and identifying trends within these large collections of data [11]. These techniques can "learn" to identify these patterns from the data they are given assuming the data is properly processed and presented. The work presented within this thesis primarily falls into the supervised machine learning which can be further broken down into additional, discrete algorithms.

Supervised Machine Learning is a process by which a model is developed by feeding it data that has been labeled. The label indicates the outcome the model is intended to predict [12]. It goes through a series of mathematical constructs such as back propagation and gradient descent within deep learning to learn the function that best models the pattern or trend within data which is then tested against this label. The closer the function and model get to correctly discerning the label, the more accurate and effective the model becomes.

Classification is when a model is able to classify a batch of input variables to a discrete output called classes [13]. There are various forms of classification such as binary or multi-class. Classification is utilized within this thesis, such as when a model is attempting to detect a fall from an individual. The labels in this instance correspond to whether a person has fallen or not fallen. The metrics for such a use case compare how many times a model accurately determined

the correct class or label in comparison to how many times it was incorrect. Models are never perfect, but a good model will be accurate a large majority of the time.

Regression is utilized when a model is attempting to predict a potential range of continuous outputs [14]. This can be more easily represented by trend lines in traditional statistical analysis. However, machine learning and deep learning can take it much further and can do so with fewer data processing ahead of time, especially in the case of deep learning. This is utilized within this thesis as well and within multiple places. The human pose estimation model and the time series forecasting models are both regression-based models that predict multiple values rather than discrete values. The metrics for this type of supervised learning involve comparing the output of the model to the label(s) given. There are a number of metrics represented here with an example being Mean Absolute Error. These metrics will be discussed at length further within the thesis.

Algorithms

The research presented within this thesis utilized a number of specific algorithms in order to perform supervised machine learning.

Deep Learning

The algorithms within this thesis fall primarily within the deep learning field of AI. These algorithms utilize artificial Neural Networks in order to determine the function that best fits the data and trends within the data. This was, at a high level, a method by which to imitate the human brain and its neuronal system [11]. It utilizes certain mathematical constructs such as back propagation and gradient descent in order to propagate what it has learned from its attempts, back through the neuronal system adjusting parameters known as biases and weights as it goes.

These systems have a few inherent advantages. In comparison to standard machine learning algorithms, certain data preparation steps are unnecessary such as feature selection. These models, through the mathematical constructs and neuronal nodes inherent to the model, will

determine which features of the data are valuable for making an accurate prediction. In addition, due to the inherent complexity that can be created with these models, they can be more accurate than traditional machine learning, especially with complex and large datasets.

However, there are a few drawbacks. While they can reduce and improve the data preprocessing in some manners, they require adjustment in others. As is discussed further down, depending on the deep learning model, the data has to be in a certain shape. This shape is the method by which data is fed into the deep learning model. In addition, there is a common saying within deep learning and machine learning as a whole which is "garbage in, garbage out". This refers to the concept that the model is only as good as the data it is fed. As such, deep learning is extremely data reliant. They work better with large datasets and more information. They will struggle with smaller datasets to find the trends and patterns within the data appropriately. In addition, due to their ability and preference for large datasets, they are also computationally expensive. This can be ameliorated by increased computing power, but it is still a cost of using these systems and models.

Temporal Deep Learning

Going further into the deep learning field is the need for temporal deep learning meaning deep learning that accounts for and appropriately handles time. Early deep learning models, while effective, did not account for elements such as time. This led to the invention of the recurrent neural network (RNN). This is a deep learning model whereby the artificial neurons are connected to each other to form a directed or undirected graph in a temporal fashion [15]. This allows it to better model and predict temporal sequences. However, these models have a weakness in that they will forget information if it has happened considerably long ago within the temporal sequence known as the vanishing gradient issue [16]. An example of this is using an RNN to decipher sentences and writing. If a reference has happened much earlier in a paragraph or sentence, then the model will fail to properly identify this behavior.

As you can imagine, this was a highlighted problem within the community and was solved with new deep learning models – the long short-term memory (LSTM) model and the Gated

Recurrent Unit (GRU) model [17] [18]. Both models utilize similar behaviors of extending the neuronal nodes to be more complex and have the ability to remember previous information for longer periods of time. This helped these models to be significantly more accurate over time than plain RNNs. Despite this improvement, they are not perfect and will still forget information if it has not been useful soon enough. In addition, as stated above, these models and RNNs, in general, require the data to be in a specific 3D shape. This is represented by input needing to be represented as such: batch size, time steps, sequence length. Batch size does not have to be solved within modern-day libraries as it is calculated by the time steps and sequence length. However, the 3D shaping of data makes working with these models more difficult and needs to be considered.

Time Series Forecasting

Time Series Forecasting is utilizing deep learning or other machine learning techniques to predict what will occur next within a time series of presented data [19] [15]. As is implied by the name, this is a very temporally oriented process and benefits heavily from models that have a temporal nature such as the aforementioned LSTM and GRU. The most obvious use case of this algorithm and process is to predict what the weather may be within a certain period of time in the future. The LSTM model can be trained to determine what the output would be based on a collection or batch of previous time points. This methodology is also used heavily when attempting to predict what might happen in the stock market. However, these models are notoriously difficult to get accurate as they require certain criteria that make it difficult to predict outlier situations that may and often do occur within the above scenarios. These criteria are explained within this thesis within the preprocessing portion of the methods sections.

Related Work and Literature Review

Human Pose Estimation – Literature Review

Human pose estimation was originally a traditional machine learning problem. However, this method was often ineffective at solving the problem. As such, along with many aspects within topics of computer science, the advent of deep learning provided a new method by which to solve the problem at hand. This was first taken in the paper of DeepPose: Human Pose Estimation via Deep Neural Networks [20]. This paper was the first attempt to use deep learning and specifically CNNs or convolutional neural networks within deep neural networks to extract useful features and estimate human poses. This paper introduced the CNN and used cascaded DNNs to refine joint positions and make better estimates of where the joints would be. The combination of these techniques created a model which produced State of the Art results at that time. In addition, it eliminated the need to create handcrafted features as is common within standard machine learning methods.

Tompson et al. took this work further though in a more general way [21]. One of the problems with the aforementioned work within DeepPose was the use of regression for joint locations. While this may seem at first to be a good idea, there are a number of limitations. CNNs are not ideal when producing single-digit outcomes such as x and y locations of joints within poses. In addition, pooling layers are often used to reduce spatial invariance. However, in the process of doing so, spatial resolution is reduced and accuracy decreases. To resolve this issue, Tompson et al. replaced direct joint regression with the use of coarse heat maps. Essentially, they use multiple resolutions of the image within the ConvNets and produce a likelihood that a joint is at an x, y location, a heatmap. A second model called the fine heat-map model evaluates these positions after taking the max likelihood locations and further refines joint locations to a final position.

Newell et al. introduced a new model architecture called the stacked hourglass network for pose estimation [22]. In essence, it takes the original image and downsamples it as is normal from pooling layers. However, it then upsamples back and is supervised at every level. The

advantage of this is that both global and local information is captured. Local information in this context is referring to elements such as the shoulder or wrist joints. Networks often crop down to joints for more accuracy in predicting joints. However, global context is needed when producing a pose estimation outcome. By having good resolution at both local levels and global levels, accuracy is increased.

Sun et al. took this concept further with the HRNet or the High-Resolution Network [23]. This was a very simple but highly elegant idea in which the previous methodology of going from a high-resolution representation to a low-resolution representation back up to high-resolution representation was replaced. This architecture still existed, however, instead of losing the original high-resolution representation, it was persisted and fused along with low-level representations through the architecture. This maintenance of high-resolution representations throughout the network leads to significantly increased performance.

Chu et al. introduced multi-context attention for pose estimation [24]. This introduces multiple novel features starting with an extension of the Stacked hourglass network. This paper introduced the Hourglass residual unit for increasing the receptive field of the model and subsequently, the model learns features within multiple scales. However, the large improvement introduced by this paper was the use of a visual attention mechanism and applying that mechanism in a multi-context fashion. This means applying the mechanism to each individual joint or body part in addition to the entire human body. Performing the work this way increases the accuracy of the heat map for each body part.

Artacho et al. introduced Unipose,a powerful top-down pose estimation model. Unipose combined a number of novel methodologies into a single model to get state-of-the-art results. The most notable introduction was the waterfall atrous spatial pooling (WASP) module. This combined 3 previous elements within machine learning research into a singular module most notably: ASPP, Cascade, and Res2Net. This module obtains multi-scale features but in a way that reduces parameter size and saves on memory constraints. The aforementioned multi-context attention work while effective was very demanding on computational resources. The use of the WASP module attains similar results with less computational complexity.

Artacho et al. pushed their own research further with Omnipose which took their previously mentioned Unipose work and applied it to multi-person pose estimation [25]. This work showed how powerful their previously used WASP module is and in this work, they extend it and make it more powerful while applying it to multi-person pose estimation. In addition, the paper introduces a "lite" version of their model which takes inspiration from MobileNet as a means by which to reduce the number of parameters and subsequently, memory requirements of the model. This is an important step in taking human pose estimation and making it truly useable as it often requires significant computational resources.

Cao et al. introduced the powerful OpenPose system, utilizing part affinity fields for fast and general pose estimation [6]. This paper utilizes a bottom-up approach as opposed to the top-down approach of Unipose and Omnipose. This allows the model to create key points for every individual within a scene and then associate these key points together via the part affinity field. While this system may not have state-of-the-art performance listed, it is much more powerful as a general pose estimation model. For this reason, greater success was had with this model in comparison to Unipose.

Lead Author	Description	Dataset	Methods	Accuracy
Toshev et al.[20]	Examined if human pose estimation could be solved via deep learning instead of machine learning, specifically using Convolutional Neural Networks	Frames Labeled in Cinema ~1000 Testing Images ~4000 Training Images Leeds Sports Dataset ~1000 Testing Images ~11000 Training Images	Convolutional Neural Network	PCP ~70%
Tompson et al.[21]	Examined and pioneered the use of heat maps within human pose estimation	Frames Labeled in Cinema ~1000 Testing Images ~4000 Training Images Leeds Sports Dataset ~1000 Testing Images ~11000 Training Images	Convolutional Neural Network Heatmaps	PCP ~70%
Newell et al.[22]	Examined the use of Stacked Hourglass networks which upsample the image back up and gather both global and local information	Frames Labeled in Cinema ~1000 Testing Images ~4000 Training Images MPII Human Pose ~11000 Testing Images ~28000 Training Images	Convolutional Neural Network Stacked Hourglass Heatmaps	PCKh ~90
Sun et al.[23]	Examined the creation and use of the High Resolution Network or HRNet. This worked similarly to the Stacked Hourglass but maintained the original high-resolution image throughout the model	COCO ~25,000 Testing Images ~57,000 Training Images	Convolutional Neural Network Heatmaps high-resolution Network	PCKh ~92.3

Chu et al.[24]	Examined and pioneered the use of the visual attention mechanism	Leed Sports Dataset ~1000 Testing Images ~11000 Training Images MPII Human Pose ~11000 Testing Images ~28000 Training Images	Nested Hourglass Network Heatmaps Multi-Context Attention	PCKh ~92.6
Artacho et al.[26]	Examined and introduced the use of the WASP module as well as incorporating everything into a singular flow	Leed Sports Dataset ~1000 Testing Images ~11000 Training Images MPII Human Pose ~11000 Testing Images ~28000 Training Images Penn Action ~2300 video sequences BBC Pose ~ 309,000 Testing Images ~610,000 Training Images	WASP module Heatmaps LSTM (In video version)	PCKh LSP ~94.5% MPII ~92.7% Penn Action ~99.3%
Artacho et al.[25]	Examined and introduced the use of the WASPv2 module as well as making their previous work effective for multiple- person contexts	COCO ~ 250,000 images MPII ~25,000 images	WASPv2 Heatmaps high-resolution Networks	PCKh MPII ~92.3% COCO ~81.2%
Cao et al.[6]	Developed the bottom-up and generally applicable OpenPose system for multi-person human pose estimation via part affinity fields	COCO ~ 250,000 images MPII Human Pose ~11000 Testing Images ~28000 Training Images	Part Affinity Field Heatmaps Convolutional Neural Network	PCKh MPII ~79.1% COCO ~71.3%

Table 1: Summary of Literature Review – Pose Estimation

Fall Detection – Literature Review

Visual fall detection is a more niche field in comparison to the aforementioned pose estimation field. The fall detection field in general is heavily occupied by wearable fall detection such as those that utilize the apple watch. In addition, a large number of papers prefer to utilize depth cameras due to privacy concerns. While these concerns are understandable, depth cameras are quite rare in comparison to normal RGB cameras. RGB cameras are now in abundance due to the advent and popularity of the smart phone. Below is a literature review of fall detection and prediction papers that provided useful information for the organization of this thesis.

Nunez-Marco et al. is an earlier paper that helped begin the view of deep learning within vision-based fall detection [27]. It is one of the earlier papers detailing the usage of a CNN as a method to detect falls within videos and images.

Kong et al. is a more modern paper that looks at various elements that had not previously been addressed within the fall detection sphere [28]. A few key elements they looked at were the impact of lighting on fall detection effectiveness and also, most notably, camera height. They compared camera height along with a number of image detection and recognition models to try and best determine the effectiveness of various methodologies.

Iazzi et al. showed and compared various methods of background subtraction along with posture representation [29]. They first subtracted the background from various scenes, comparing background subtraction methods along the way. This pre-processing step allows for better usage of posture representation. Posture representation is used as a way of taking the shape of a human and evaluating that based on a histogram. Note that posture representation and human pose estimation are separate and unrelated endeavors. Lastly, once the posture representation has been produced, fall detection is enacted on the output histogram of posture representation to detect falls.

Chhetri et al. is a modern, deep-learning-based, vision fall detection system [30]. It uses CNNs as a method to detect falls within images and video based on an optical flow methodology. While I plan to use human pose estimation for my pre-processing step, this paper noted some difficulties within fall detection and brought focus to the need for attention within the processing aspect of these systems. Oftentimes, the lack of focus and understanding of the processing needed for fall detection leads to systems and research that would be impossible to implement.

Wang et al. is a modern paper that closest approximates my goals [31]. This paper uses OpenPose, the same system this thesis devised to use, to produce key points. It produces state representations from these key points which can then be used to detect falls. It notably uses some other representations such as centroid detection and uses the bounding boxes within its fall detection methods.

Lead Author	Description	Dataset	Methods	Accuracy
Nunez-Marco et al [27]	One of the earlier papers on using CNN's as a method of fall detection within vision-based systems. This paper was primarily used as a guiding light for how fall detection with vision-based systems could be implemented in a simpler manner from a neural network standpoint. It uses a basic CNN along with optical flow and transfer learning to detect falls or no falls within datasets.	UR Fall (URFD) ~11,900 Total frames ~900 Fall Frames ~30 Falls ~900 No Falls Fall Detection Dataset (FDD) ~260,000 Total Frames ~7,800 Fall Frames ~184 Falls ~350 No Falls (Multicam) ~ Multiple Camera Fall Detection 100,000 Total Frames ~4,000 Fall Frames ~127 Falls ~200 No Falls	Convolutional Neural Network Optical Flow Transfer Learning	Sensitivity URFD: 100.0% FDD: 98.07% Multicam: 93.47% Specificity URFD: 94.86% FDD: 96.20% Multicam: 97.23%
Kong et al. [28]	This paper addresses and looks at various elements that have not been previously evaluated within fall detection such as camera height. They also detail other limitations that have been proffered within the field such as lighting level. In addition, it points out limitations in previous skeleton-based simulations which rely on the Microsoft Kinect. Their solution is to use a tracking and denoising Alex-Net model prior to fall detection. In addition, this study looks at camera height for the effectiveness of the models.	Fall Detection Dataset (FDD) ~260,000 Total Frames ~7,800 Fall Frames ~184 Falls ~350 No Falls	Depth Camera Human Segmentation AlexNet ETDA-Net LENet HOG+SVM Convolutional Neural Network	Best Camera Height: 2.1 Meters Best Specificity: EDTA-Net: 100% Best Sensitivity: HDTA-Net: 100%
Iazzi et al.[29]	This method uses a background subtraction preprocessing step. This step removes the background such that the individual can be evaluated better. It then uses Human Posture Representation in order to extract the human shape. Note that this is not the same as human pose estimation as its output is shape-based and requires the Microsoft Kinect. They then use posture classification followed by fall detection in order to attempt to detect falls.	UR Fall (URFD) ~11,900 Total frames ~900 Fall Frames ~30 Falls ~900 No Falls Fall Detection Dataset (FDD) ~260,000 Total Frames ~7,800 Fall Frames ~184 Falls ~350 No Falls Multiple Camera Fall Detection (Multicam) ~100,000 Total Frames ~4,000 Fall Frames ~127 Falls ~200 No Falls	Background Subtraction: Various Methods (GMM,AMF,CB) Projection Histogram Posture Classification: Various Methods (SVM,KNN,DT,NN etc.)	Accuracy: 97.29% Precision: 97.48% Recall: 97.48
Chhetri et al. [30]	This paper is a modern paper focusing on fall detection using deep learning and CNN's as a method for detecting falls. It also mentions a great deal about processing time, a component that is very important for the combination of fall detection and Human Pose Estimation.	UR Fall (URFD) ~11,900 Total frames ~900 Fall Frames ~30 Falls ~900 No Falls Fall Detection Dataset (FDD) ~260,000 Total Frames ~7,800 Fall Frames ~184 Falls ~350 No Falls Multiple Camera Fall Detection (Multicam) ~100,000 Total Frames ~4,000 Fall Frames ~127 Falls ~200 No Falls	Convolutional Neural Network Enhanced Dynamic Optical Flow	Probability Score: URFD: ~93% FDD: ~90% Multicam: ~92%
Wang et al.[31]	This paper is the closest to what I am looking to do as well. It uses Openpose to produce human pose estimation markers and then does fall detection. The preprocessing step uses Openpose to produce body key points. Once the key points are created, they use various machine learning methods and state representations to attempt to detect falls. There is also a marked attempt to be wary of processing time and power.	UR Fall (URFD) ~11,900 Total frames ~900 Fall Frames ~30 Falls ~900 No Falls Fall Detection Dataset (FDD) ~260,000 Total Frames ~7,800 Fall Frames ~184 Falls ~350 No Falls Le2i Fall Detection Dataset ~191 human activity videos in four scenes	Preprocessing Open Pose Deeper Cut Fall Detection: Various methods (Random Forest, SVM, KNN, MLP)	F-Score URFD: 97.78 Le2i: 97.08

Table 2: Summary of Literature Review – Fall Detection and Prediction

Dataset

Fall Detection Dataset

For the work being performed within this experiment, a dataset consisting of images depicting individuals falling and not falling was needed. These instances of falling and not falling needed to be labeled. As such, the UR-Fall dataset was chosen [5]. This dataset consists of 70 videos with manual labels for not fall (-1), fall (1), and a transition label (0). 30 of these videos depict falls and the other 40 depict activities of daily life (ADLs) whereby the individual does not fall within the video. After processing, this provided approximately 11,000 rows of data. It should also be noted that the dataset itself proposes to note utilize the transition label as it makes detection significantly more difficult.

Key Point Dataset

From the above dataset, a new dataset was created. Utilizing the aforementioned human pose estimation, key points were output from each image for the joint locations of the human within the image. After some processing which is further described below, this led to a dataset with 36 key points. This number comes from the number of key point locations output from the human pose estimation system, 18, multiplied by two as the x and y locations need to be split for the models to be used further down in the process.

Feature Number	Feature Name
1	nose_x
2	nose_y
3	neck_x
4	neck_y
5	Rsho_x
6	Rsho_y
7	Relb_x
8	Relb_y
9	Rwri_x
10	Rwri_y
11	Lsho_x
12	Lsho_y

13	Lelb_x
14	Lelb_y
15	Lwri_x
16	Lwri_y
17	Rhip_x
18	Rhip_y
19	Rkne_x
20	Rkne_y
21	Rank_x
22	Rank_y
23	Lhip_x
24	Lhip_y
25	Lkne_x
26	Lkne_y
27	Lank_x
28	Lank_y
29	Leye_x
30	Leye_y
31	Reye_x
32	Reye_y
33	Lear_x
34	Lear_y
35	Rear_x
36	Rear_y

Table 3: Summary of Key Points

Feature Number	Feature Name
1	Angle_1
2	Angle_2
3	Angle_3
4	Angle_4
5	Angle_5
6	Angle_6
7	Angle_7
8	Angle_8
9	Angle_9
10	Angle_10
11	Angle_11
12	Angle_12
13	Angle_13
14	Angle_14
15	Angle_15
16	Angle_16

Table 4: Summary of Angles

Angles

After the creation of this temporary dataset, angles can be calculated via the joints. The process for this is further described below. Once the angles have been calculated, they are concatenated onto the dataset. The angles are simply named 1 through 16. Lastly, the labels of the falls can be concatenated onto the end of the dataset to create the completed dataset containing key point locations in x and y, angles, and the fall label itself.

Methods

Overview

This section contains the experiments that were performed and the reasoning behind them. Additionally, it provides more granular information about what algorithms are used, their configurations, preprocessing of data that took place, the dataset(s) used, as well as metrics that were evaluated to determine the effectiveness of the models themselves.

Experimental Design

Following the literature review and analysis of related works, experiments were conducted in order to answer the research questions laid forth previously. First, a human pose estimation system is established in order to provide the proper features within which to conduct the experiment. The benefits of such a system are that they are designed to provide estimations through obstructions and changing light and shadow within a scene. The system also scales down our problem from one of a complete computer vision problem into a time series problem whereby we have rows of data for features such as key points of the joint locations. This model is utilized against a computer vision-based fall detection dataset that has had "falls" and "not falls" manually labeled. After running the model, the falling label can be concatenated onto the dataset such that we can predict on the key points and not the visual images anymore. This accomplishes a few objectives of the thesis including the creation of a new dataset with the key points.

Next, a bespoke fall detection model is built to assess whether falls can be detected from the key points and any features created from them. The key points can be utilized to create new features such as angle calculations between the joints. This system is then tested using the new dataset that was created to determine whether falls can be detected from key points and angles alone. It is important to remember that at this point, the data has been heavily paired down to

rows of integers, drastically reducing the size of the problem and simplifying the issue at hand. However, this also means less information is available to work with.

Third, a bespoke model is built for time series forecasting. This model is able to look at a few rows of the new dataset and can estimate what the key points and angles would be for the next row. This can then be run over the entire dataset to create a new dataset, one that has only been estimated from the previous rows. By doing this, the model is attempting to estimate and predict one row into the future. This can then be concatenated back with the labels from the dataset.

We now have all of the pieces needed. This last, future estimated dataset can be run against the fall detection algorithm. Due to the fact that we are estimating at least one row into the future and detecting falls on this predicted row, we are predicting falls occurring before they actually occur by a single frame. The intention is for this model to start with one row of predictions to show that it is feasible. However, this could be theoretically pushed further for longer-term prediction.

Human Pose Estimation

The first step within the experiment was to set up and utilize a human pose estimation system. Due to the focus being on fall detection and prediction as well as the results of the literature review, it was decided to utilize an established model for this effort. In this instance, the model utilized was OpenPose. OpenPose is a bottom-up, human pose estimation system that generates key points of the locations of joints within an image and returns them as an output for 18 joints [6] [7] [8] [9]. Bottom-up in this context means that the system places all of the key points without first segmenting out the humans within the image via a bounding box. This system is well-established and provides a good benchmark with which to start. It does have a few minor hiccups such as attempting to place key points on elements of the image that are not humans such as chair joints. However, OpenPose also utilizes a confidence score for each of the joint calculations. Fortunately, these confidence scores are often very low for elements such as chair joints and are able to be parsed and removed down to a cleaner set.

Figure 1: Example of Pose Estimation

Angle Calculation

After the human pose estimation system has generated key points for the joint locations of a human within an image of the fall detection dataset, these key points can then be used to produce additional features. The additional feature most interesting to us was angle calculations of the joints. We had hypothesized that the joint angles may provide additional information about the likelihood to fall. This was garnered through discussions with caretakers at a senior home care facility. They had indicated that the best way to determine if someone was going to fall was by looking at the angles of their feet. This highlighted the possibility that angles could add additional information to enhance fall detection and prediction. Thus, 16 angles were calculated between various segments of joints in order to add this additional information to the

later models to utilize. These angles are detailed at greater length below within the dataset section.

Angle Calculation

Given 3 key points – a, b, and c with x and y given as ax and ay. Given that a is the fulcrum of an angle:

$$\theta = \pi 2 + (\arctan(cy - ay, cx - ax) - \arctan(by - ay, bx - ax))$$

Fall Detection

After the creation of the key points and angle calculations, fall detection needed to be performed. This determined whether it was feasibly possible to detect falls purely from key point and angle data alone. In addition, this is very temporally oriented. As such, the previously mentioned LSTM and GRU model architectures were utilized to determine whether falls could be detected and how effective they would be.

Time Series Forecasting

Now that we have a model for fall detection, we need to be able to predict out into the future. The method we used to perform this is time series forecasting. This is where we use a few past rows of information in order to estimate the values of the next row. We can then compare the actual next row to the estimated one as a form of supervised learning. Again, due to the temporal nature of this, an LSTM was utilized to perform this estimation.

Fall Prediction

All the pieces are now in place to perform the last piece of fall prediction. By utilizing the time series forecasting model, we can now attempt to detect falls on estimated future rows. This approximates fall prediction, and this process should be able to accomplish all of our objectives and answer all of our research questions.

Software Environment

Python

The coding language utilized throughout the entirety of the experiments was Python. Python is a high-level programming language that was designed for general purpose use. However, it benefits from having been one of the main languages of choice for machine learning and deep learning in recent years. As such, it contains a number of libraries that are very useful for performing the work at hand.

- Pandas: Pandas is a python library and package that provides the means to easily work with data utilized within machine learning and deep learning. It provides the DataFrame object which makes the data manipulation needed for these works considerably easier [32].
- Numpy: Numpy is an open-source package designed for scientific computing. The primary benefit of Numpy is to provide true arrays within Python which are much faster and easier to work with when performing any sort of list or array processing
  [33].
- Matplotlib: An additional python package provided mainly to produce graphs and plots for 2d representations. This package makes it easier to understand the data that is being manipulated and can highlight areas that may be of concern [34].
- Tensorflow: This is an open-source library created by Google to allow for GPUs to be utilized to address machine learning issues. It also provides a means to create models easily and quickly at a lower level [35].
- Keras: This is a deep learning API often utilized with Tensorflow. With Keras, a user no longer needs to address the low-level constructs of Tensorflow but rather can quickly create models at a high level with much fewer lines of code [36].
- Sklearn: This is another open-source machine learning library within Python that is often used with machine learning. However, it also provides useful tools that can be used within deep learning such as onehotencoding and scaling operations that are universally valuable within the AI world [37].

- os module: The os module within Python allows for easy and direct ways of addressing path locations. It is utilized when manipulating data and reading and writing data [38].
- math module: The math module within Python provides deeper mathematical operations such as square roots and trigonometry-based calculations. This module was exclusively used during the calculation of the angles between joints [39].

Computing Resources

In addition to the above resources utilized, the computing resources utilized are important to highlight. For the majority of the work performed with this experiment, the Pittsburgh Super Computing Cluster (PSC) was utilized as it provided an extensive and powerful computing infrastructure within which to perform these calculations [40]. This was provided via an allocation from the Extreme Science and Engineering Discovery Environment (XSEDE) organization [41]. The computer in specific that was utilized was the GPU-heavy, Bridges-2 computer from PSC. Shared GPU nodes were often the chosen computing platform as the system is quite powerful and dedicated GPU calculation was not necessary.

Data Preprocessing

Data preprocessing is a much-needed step within the process of producing use able data with which to work. Much of the work producing deep learning models is determining how to preprocess the data appropriately. Within this experiment, various steps of preprocessing were needed.

Human Pose Estimation

The processing for human pose estimation was the lightest throughout the experiment. It required a manual test/train split as well as appropriately manipulating the images as the images come on a per video basis. The videos have to be concatenated together in an appropriate fashion such that the model can output poses correctly.

Index	Key Point 1	Key Point 2	...	Key Point 18
0	(470,104,0,.71342),(113,251,1,.14298)	0	...	(324,275,0,.57482)

1	(468,103,0,.72831)	0	...	0
2	(466,101,0,.72153),(110,256,1,.23143)	(104,311,0,.74523)	...	0

Table 5: Example of Pose Estimation Output

Fall Detection

The preprocessing for fall detection was one of the heaviest stages of preprocessing. The output from the human pose estimation is a bit noisy and contains more than is needed for a clean dataset – one of the objectives of this research. It often contains multiple observations for a single joint in each row. It also contains a confidence score and a centroid for that joint. The goal of this research and objective is to create a clean dataset that is easy to work with. Because it is understood that the original dataset only has one human in the frame at a time, we can infer that a row should only have one reference to the joint. The confidence scores and centroids were utilized to deduce which joint to keep and which to remove as the incorrect joints often had very low confidence scores. After reducing it down to just a single joint, the centroid and confidence scores were removed. After this step, the x and y key points, which are normally within a single tuple, are separated into x and y columns of their own. This entire process is made more difficult as the output from the human pose estimation are integers, however, they are represented as strings within the .csv file that is created as the output. As such, much work is needed to adjust the values from being string values into actual integers. Due to some joints having missing estimations from OpenPose, Interpolation is needed to fill in these empty spots and improve the dataset. Sometimes the human pose estimation fails to find a joint and subsequently, the output is a tuple of zeros. Using interpolation from sklearn along with back fill and forward fill where necessary, these zeros were filled in with values that made the most sense given their location within the dataset.

Index	Key Point 1 – X	Key Point 1 – Y	...	Key Point 18 – X	Key Point 18 – Y
0	470	104	...	324	275
1	468	104	...	320	272
2	466	104	...	311	260

Table 6: Example of Output After Cleanup

Angles

After the above preprocessing, angles can now be calculated for the joints. With the dataset having had the x and y locations already split out, a function can be created for taking in 3 separately created tuples from these x and y columns and outputting an angle calculation to be utilized. This was accomplished with the math module and a simple angle calculation listed below. These results were then appended to the key point dataset along with the fall label to create the finalized dataset for fall detection. Lastly, after appending the angles, the dataset was scaled using a minmax scaler to represent the key points and angles as values between 0 and 1. These values help the LSTM and GRU models to converge much quicker and often improve the accuracy of the models as well.

Angle Number	Joints Used
1	Left Hip Left Knee Left Ankle
2	Right Hip Right Knee Right Ankle
3	Right Shoulder Right Elbow Right Wrist
4	Left Shoulder Left Elbow Left Wrist
5	Neck Right Shoulder Right Elbow
6	Neck Left Shoulder Left Elbow
7	Left Elbow Left Shoulder Right Shoulder
8	Right Elbow Right Shoulder Left Shoulder
9	Left Knee Left Hip Right Hip
10	Right Knee Right Hip Left Hip
11	Neck Right Shoulder Left Shoulder
12	Neck Left Shoulder Right Shoulder

13	Neck Right Hip Left Hip
14	Neck Left Hip Right Hip
15	Left Knee Left Hip Left Shoulder
16	Right Knee Right Hip Right Shoulder

Using the above combinations of joints, the angles can be created. Below is a table showing the dataset as it looked after preprocessing with angles added in.

Index	Key Point 1 – X	Key Point 1 – Y	...	Angle 15	Angle 16
0	470	104	...	1.1543	3.4562
1	468	104	...	1.1438	3.5671
2	466	104	...	1.1321	3.5908

Table 7: Example of Output After Angles

Lastly, after the creation of the dataset, as listed above, feature importance was calculated. While this is not needed for deep learning models, it was performed for the sake of understanding how valuable the angles were in comparison to the key points themselves. The feature importance showed that the angles provided some of the stronger additions to the dataset in terms of their importance in determining the prediction. The later values along the x-axis indicate the angles in comparison to the earlier values being key points.

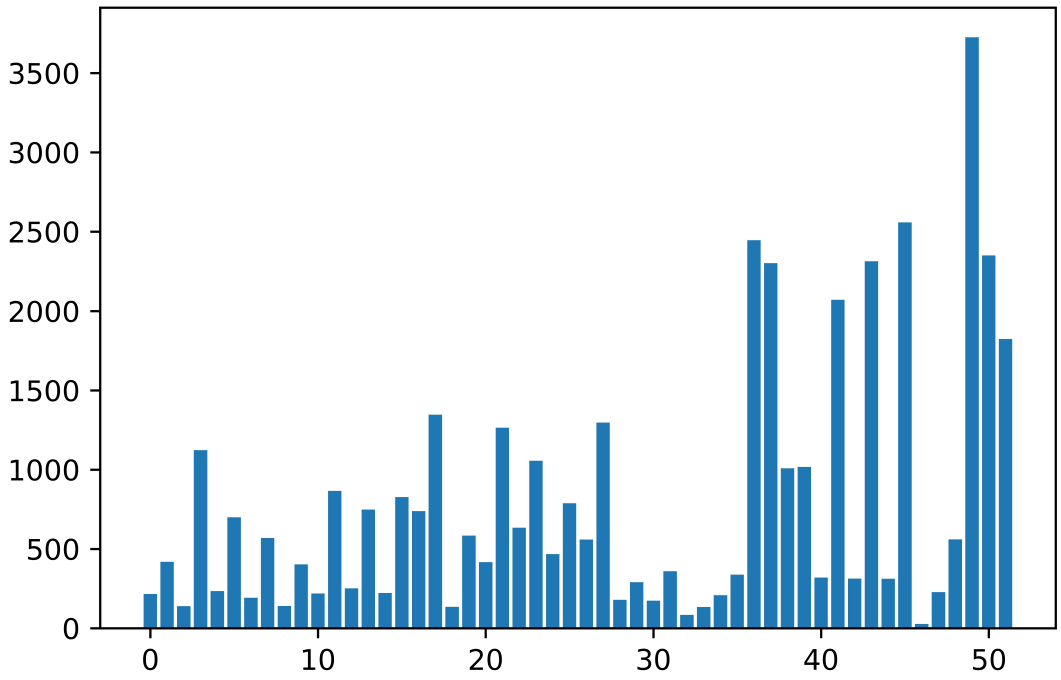

Figure 2: Feature Importance

Deep Learning Models

Now that the dataset has been cleanly created, less preprocessing is needed. However, a few changes are needed specifically for the fall detection step. The fall label, which is usually represented in -1, 0, and 1, needs to be OneHotEncoded. Sklearn was utilized in this instance to convert the 0s, -1s, and 1s to a dataframe object that contains three columns. This was utilized while including a transitory class that is labeled within the UR fall dataset. In addition, the fall detection also tried binary classification whereby the 0 was made into a not fall class. The values were also scaled to between 0 and 1, as these values work best with LSTM and GRU style models.

Sliding Window

A large aspect of preprocessing that occurs for both the fall detection and time series forecasting models is the creation of a sliding window. The sliding window is a construct

whereby multiple rows of data can be used to attempt to produce a better detection or forecast [42]. A simple example would be using rows 1,2, and 3 to detect a fall on row 3. In this context, the model gets additional, temporal information with which to utilize for making an estimation or prediction. The window is then moved forward whereby rows 2, 3, and 4 are used to detect a fall on row 4 and so on. This was created at this stage along with a function that sweeps through a window size of 0 to 20. This allowed for exploration of the effectiveness of the window size.

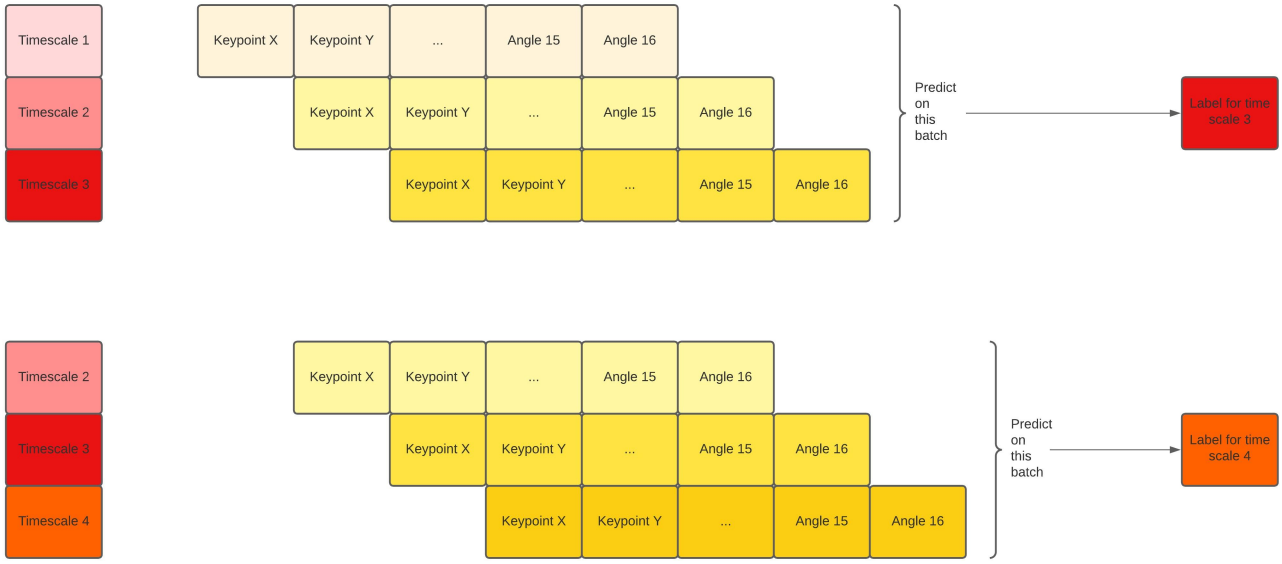

Figure 3: Example of Sliding Window

Time Series Forecasting

The preprocessing for the time series forecasting required much of the same as fall detection such as scaling and a sliding window. However, it also required some new additional processing. The primary issue with time series forecasting is the requirement of the stationality of a time series [19]. This is where the time series lacks irregular trends. This is often easiest understood within a different context such as seasonality. If data is being utilized that has an aspect of seasonality to it, this must be removed in order for the forecasting to be effective. Within our data, this was not much of an issue. However, one method utilized for the stationality proved to be vital. This is the method of difference whereby the difference between two rows is calculated [15]. This can then be forecast and added back into the original data. Essentially, this means that

the time series forecasting is not attempting to predict the next key points and angles, but rather the difference between the current key points and angles and the next ones. This difference is often smaller and easier to predict. This was combined with the sliding window to improve results as effectively as possible.

Index	Key Point X	Key Point Y	...	Angle 15	Angle 16
0	470	117	...	1.56787	3.14562
1	468	114	...	1.43241	3.04521

Table 8: Example of Rows for Difference

Index	Key Point X	Key Point Y	...	Angle 15	Angle 16
0	-4	-3	...	-0.13546	-0.10041

Table 9: Difference Generated from Rows

After the model has been trained on attempting to identify the difference between rows in order to predict more accurately out into the future, the output can be combined with each previous row to determine the actual predicted values. The below table shows what the final output of the time series forecasting looks like.

Index	Key Point 1 – X	Key Point 1 – Y	...	Angle 16
0	468.2391	103.0918	...	2.7890
1	465.0923	98.7263	...	2.6872
2	458.1082	97.2314	...	2.7123

Table 10: Example of Output from Time Series Forecast

Implementation

This section contains a high-level view of various phases of implementation that are represented above.

- Human pose estimation is run against the UR-Fall dataset to derive joint location key points from it.
- Data preprocessing takes place in order to alter the output of key points to be in a much more workable format.
- Angles are generated from the above dataset and concatenated onto the dataset.

- This new dataset is scaled and utilizes a sliding window function that sweeps through various window sizes and tests it against a GRU and LSTM for analysis of the best combinatorial output. F1 score and accuracy are compared as a basis for this process.
- Time series forecasting is used with the above dataset in order to predict and estimate the next row. Similar preprocessing takes place for fall detection with the addition of a differencing algorithm.
- Lastly, the pieces are combined to perform fall detection of future estimated rows of the time series forecasting.

Algorithm Configurations

This section contains some high-level configurations of the models mentioned previously. Multiple configurations were tested and these were the ones that provided the best output from those tested.

- Gated Recurrent Unit (GRU):

  This is the GRU for fall detection. Note that the time step size would change depending on the sweeping function.

  Layers:

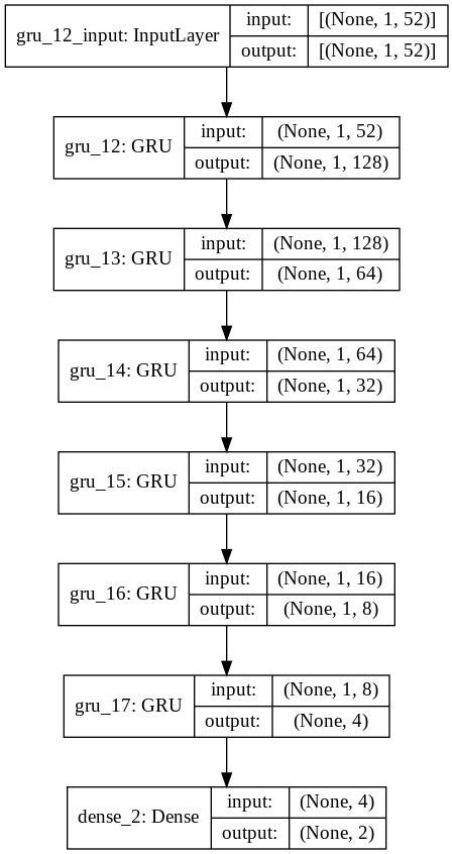

Figure 4: GRU Model

Compilation:

epochs=128

metrics = ["accuracy"]

callbacks = EarlyStopping(monitor="val_loss",patience=10, restore_best_weights=True)
model.compile(loss = "categorical_crossentropy"/ "binary_crossentropy", optimizer="adam")

- Long Short-Term Memory (LSTM):

  This is the LSTM for fall detection. Note that the time step size would change depending on the sweeping function.

  Layers:

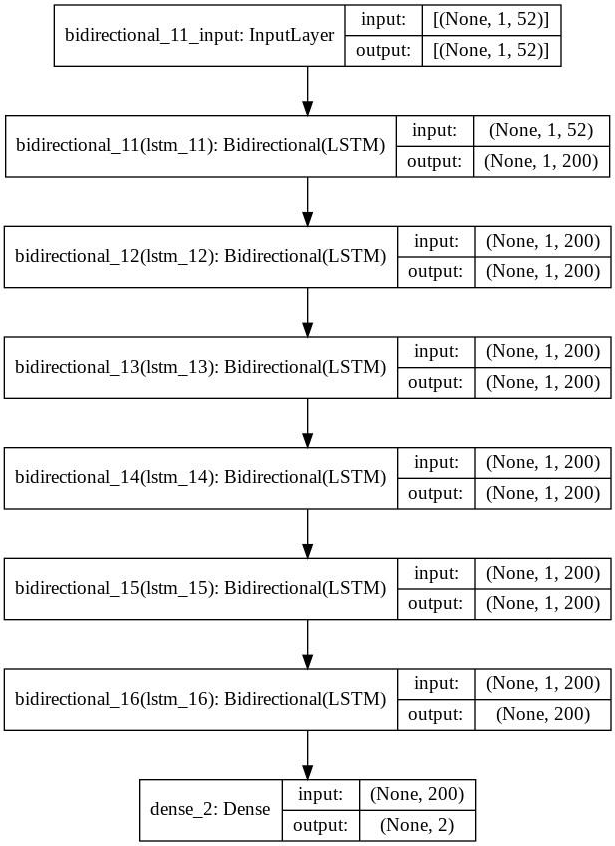

Figure 5: LSTM Model

Compilation:

epochs=128
metrics = ["accuracy"]
callbacks = EarlyStopping(monitor="val_loss",patience=10, restore_best_weights=True)
model.compile(loss = "categorical_crossentropy"/ "binary_crossentropy", optimizer="adam")

- Long Short-Term Memory (LSTM):

This is the LSTM for Time Series Forecasting.

Layers:

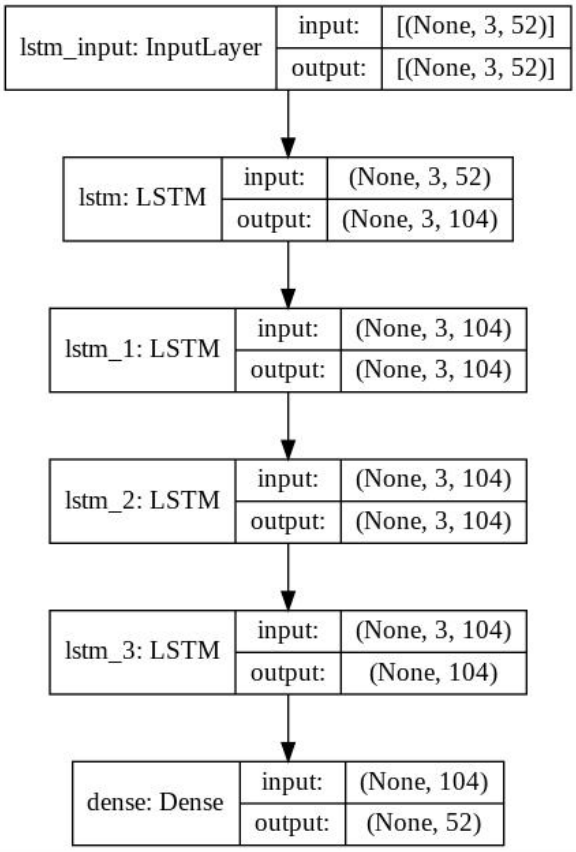

Figure 6: LSTM Model

Compilation:

epochs=10 metrics = ["mse"]

callbacks = EarlyStopping(monitor="val_loss",patience=5, restore_best_weights=True)

model.compile(loss = "mse", optimizer="adam")

Performance Metrics

An essential task of machine learning and deep learning is gathering an understanding of how effective the algorithms being utilized are. This is done through the usage of metrics. Metrics can be used to quickly evaluate the effectiveness of a model.

Classification Metrics

Classification metrics often revolve around utilizing a confusion matrix and determining the accuracy and, more effectively, the F1 score [43]. A confusion matrix is a matrix that is utilized to compare the values between what is predicted and what is the ground truth. Utilizing the confusion matrix along with certain metrics below, the F1 score can be calculated.

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

$$\text{F1} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

Regression Metrics

Similarly, regression metrics are needed to determine the performance of a regression-based model. In this instance, this applies solely to the time series forecasting model as this model is attempting to estimate a theoretically infinite range of continuous values. The primary metrics utilized here are mean squared error (MSE) and mean absolute error (MAE) [43].

$$\text{Mean Absolute Error} = \sum_{i=1}^{D} |x_i - y_i|$$

$$\text{Mean Squared Error} = \sum_{i=1}^{D} (x_i - y_i)^2$$

Results

Experiment Results

This section contains the results that were obtained from the experiments previously laid out. It is broken into sections based on the purpose and models used.

Fall Detection

The first results to analyze are the fall detection results. As a reminder, this is attempting to predict falls from the key points and angles derived from the key points. The models take in a scaled calculation of the key points and are swept across various sliding windows from 0 to 20 to find the best possible combination of window size in combination with model evaluation.

GRU results

The GRU was quickly found to be not as accurate as the LSTM, however, it was often more performant even when utilizing similar unit sizes and stacked layers. As such, it was used as a quick analysis for any changes made to models or the data before running it on the larger and slower LSTM which provided better accuracy. Below is the confusion matrix along with the calculated F1 score. The most important score listed here is the macro F1 which is the F1 for the predictions as a whole.

Label	Precision	Recall	F1-score
0	0.87	0.92	0.89
1	0.81	0.73	0.77
Micro Avg	0.85	0.85	0.85
Macro Avg	0.84	0.82	0.83
Weighted Ave	0.85	0.85	0.85
Samples Avg	0.85	0.85	0.85

Table 11: GRU Results

Label	Precision	Recall	F1-score
0	0.96	0.91	0.93
1	0.77	0.89	0.83
Micro Avg	0.90	0.90	0.90
Macro Avg	0.87	0.90	0.88
Weighted Ave	0.91	0.90	0.91
Samples Avg	0.90	0.90	0.90

Table 12: LSTM Results

LSTM results

The LSTM, while not as performant as the GRU, provided higher accuracy and F1 scores. It was stacked larger and had larger units, though it should be noted that this is after experimentation on the model on what works best. Below is the listed F1 score and other metrics from the best model found from running the sweeping function on a manual train and test split.

As shown above, the best F1 score attained was .86. This is a fairly high F1 score and shows that fall detection from key points and angles is possible. It should be noted that this is the binary classification attempt. The best multi-class classification whereby we are attempting to predict the in between falling and not falling class is listed below. Note that the window for this score was a lag period of 5. This means that the model was taking in 5 rows to detect the fall on the last row analyzed of those 5.

The best performance as listed for multi-class classification is a .76 F1 score. This was notably harder for the model; however, it was quickly discovered that this is likely unnecessary for fall detection. As such, with the better F1 score and likely unnecessary need of the intermediate label, it was decided that for fall prediction that binary classification would be enough.

Angle Benefit

One of the questions we wished to answer with our research was whether the angles added additional performance to the detection of falls. Below is the results without the angles utilizing the best LSTM configuration listed above.

Label	Precision	Recall	F1-score
0	0.75	0.91	0.82
1	0.82	0.59	0.68
Micro Avg	0.77	0.77	0.77
Macro Avg	0.79	0.75	0.75
Weighted Ave	0.78	0.77	0.76
Samples Avg	0.77	0.77	0.77

Table 13: Without Angles

As shown, this model does fairly well. However, in comparison to the best of the LSTM with angles, it produces worse results. In addition, it should be noted that more importantly, the model does better at detecting falls, the 1 label, better when angles are present. Lastly, the general performance was always around .10 higher in F1 score when comparing the two datasets and the models produced from them.

Fall Prediction

The fall prediction results are really the combination of the capability of the time series forecasting model and the fall detection model. The better the performance of these two models, the better the fall prediction capability will be. Below are the listed performances for the time series model and the entire system as assembled.

Time Series Forecasting

The time series forecasting model, unlike the previously noted models, is a regression-oriented problem. It is attempting to estimate and predict the next row of data from the preceding 3 rows of data. Due to the model being regression-oriented, the metric for accuracy is different as was listed above in the metrics section. In addition, it is difficult to gather the accuracy of the entire model over the course of the running into a single number. In order to create this singular number and also have it be as understandable at a glance as possible, it was decided that Mean Absolute Error would be a good basis. This is calculated by taking the difference between the prediction and the actual row it was attempting to predict. For instance, if the model predicted an x value of the right shoulder joint to be 308 and the actual number was 305, then the MAE is 3. This is then added up for each feature, a total of 52, and then divided by that number in order to get a singular value for the accuracy of prediction. However, this just gives us how accurate the model is for a singular row. These values then need to be added up for each row and then divided by the number of rows for an overall MAE of the entire model over the course of predicting the next row for the entire dataset. Performing the above operation gives us the following MAE.

Data	Mean Absolute Error

First Row	2.1712233816117825
Entire Dataset Average	3.4767067047800175

Table 14: Time Series Forecasting Results

Visually analyzing the results, these did seem to be fairly accurate to the truth that they were estimating. Often off by only a few points in either direction. These numbers indicate a fairly low variance between the prediction and the actual row though there was indeed a difference.

Future Falling Prediction

Finally, putting together the time series forecasting along with the fall prediction LSTM model to analyze what level of performance we get when trying to determine whether a fall has happened. As a reminder, this is utilizing the best performance LSTM model on the dataset created via the time series forecasting whereby it is predicting one row ahead. The results are as follows:

Label	Precision	Recall	F1-score
0	1.00	0.85	0.92
1	0.33	0.95	0.49
Micro Avg	0.86	0.89	0.86
Macro Avg	0.66	0.90	0.70
Weighted Ave	0.95	0.89	0.89
Samples Avg	0.86	0.89	0.86

Table 15: Fall Prediction – Best Fall Model

This model shows that while falls can be predicted, it immediately drops the F1 score quite a bit, approximately 20 points at the macro level. In addition, the detection of a fall, represented by the 1 label, is much harder to predict when the time series has been forecast.

The Oddball

As one final curveball within this process, it was decided we would run some of the lower performance models on the time series forecast. A very interesting result occurred. One of the lower F1 score models we had produced while sweeping through the window values ended up having a much higher performance on the time series model. This model had a macro F1 score of 77 – .11 lower than our highest value. However, its metrics for the time series prediction are listed below and indicate much better results.

Label	Precision	Recall	F1-score
0	0.94	0.91	0.92
1	0.65	0.72	0.68
Micro Avg	0.88	0.88	0.88
Macro Avg	0.79	0.82	0.80
Weighted Ave	0.88	0.88	0.88
Samples Avg	0.88	0.88	0.88

Table 16: The Oddball

As shown above, the results are surprisingly good, much better than the higher performance model. The overall macro score is .10 higher. However, and more importantly, the F1 score for the fall prediction is almost .20 higher. This indicates this model is much better at predicting falls. Visually observing the confusion matrix that came from this result indicated it predicted more than twice as many falls. I believe this may have come from the window shape. The model with worse fall detection performance was a 0 lag model, indicating no window was taken into account. This is in contrast to most of the models typically doing better with more rows. However, perhaps with time series forecasted rows, the additional information creates problems or confusion from built-up errors from the forecasting of the rows themselves.

Analysis and Discussion

Analysis of Experiment

The experiment was conducted in a number of phases:

- Evaluation of fall detection and prediction through literature review to answer the feasibility of the work.
- Evaluation of human pose estimation in order to determine possible output and outcomes from the usage of such a system.
- The usage of human pose estimation system to create key points for joint locations in order to create an abstracted dataset for fall detection based on visual falls.
- The creation of a fall detection model that utilizes the key points and angles created from them for fall detection.
- The usage of time series forecast to estimate the next row of data from 3 rows of data in order to create a new, future forecasted dataset upon which to attempt to predict falls.
- Lastly, the usage of the fall detection model on this new time series forecasted dataset in order to analyze the possibility of predicting falls before they occur.

Human Pose Estimation

The human pose estimation literature study showed that pose estimation had become generally capable of being applied to images with humans. This was demonstrated effectively using the human pose estimation model, OpenPose. This model, when applied to the UR-Fall dataset, was able to consistently create accurate key point locations for joints to further other experiments within this thesis

Fall Detection Experiment

After the creation of the key points via OpenPose, fall detection was performed using the labels from the UR-Fall dataset and an LSTM model. This experiment provided the highest F1 score of .88, showing that fall detection from pose estimations was possible with a relatively high degree of accuracy.

Forecasting Experiment

Extending on the previous work, it was questioned whether a new dataset of key points and angles could be created by predicting the next row of data from a preceding set of rows. This is known as time series forecasting. This experiment was able to be conducted such that the total MAE for the dataset created was, on average, approximately 3.5. This is a relatively low MAE, showing that the prediction of the next row of data from the preceding 3 rows was a possibility.

Fall Prediction Experiment

Lastly, the fall prediction experiment sought to combine the previous endeavors into a single system for predicting falls ahead of their occurrence. This was done by taking the time series forecasting dataset of predicted rows and running the fall detection model against these rows and features. This model also was able to produce a decent F1 score of .80. However, this model was noticeable weaker in predicting the falling label with an F1 score on that specific label of .68. This highlights that it is most likely possible to due future falling prediction and estimation, however, it is susceptible to accuracy loss.

Discussion

This section is a discussion based on the objectives and research questions initially posed by this thesis.

RQ1: Can the output of a pose estimation network be used to detect falls?

The outcome of this question was a positive one, whereby it was shown that this can be done to a fairly high degree of accuracy. As listed, a number of methodologies can improve this

outcome. It is also most likely that a larger dataset would improve the general performance of the model as a whole. The dataset was very useful for providing hand labeled falls, however, it is not a particularly large dataset. In this regard, a difficulty within the fall detection space is the lack of a large dataset with significant amounts of labeled images for falls.

RQ2: Can the calculation of angles from key points of a pose estimation network aid in the detection and prediction of falls?

The angle calculations performed did help the performance of the model by a maximum of approximately .10 in the f1 score. This is a fairly strong increase. In addition, while not needed for deep learning models, feature importance was performed as an exploratory analysis. This feature importance showed a few of the angles to be the most correlated with the label. A drawback of this is that the angles were hand created and picked. We hypothesize that there are angles that are most likely odd combinations that would not be chosen that may provide significant value and increased information to the model with which to perform fall detection.

RQ3: Can the output of a pose estimation network be time series forecast to predict the next row or rows that would be output?

One of the key elements of this experiment is being able to time series forecast the rows of data we have into the future and predict what the next row may be. This was a more challenging element and required the use of various techniques to improve the result. In the end, the result was fairly close to the actual rows but as expected was not perfect. That being said, we believe our work showed that it was possible to predict the next row with a decent level of precision. However, this is also the area in which there is more room to explore and more performance to be found.

RQ4: Can the combination of time series forecasting and fall detection on key points from pose estimation predict falls?

The combination of the above efforts has led to the ultimate goal of this work in attempting to predict falls before they occur. While only one row ahead at this juncture, we believe that our work has shown that it is possible to accurately perform this work. However, it would need more exploration in order to improve the accuracy to a higher degree.

Conclusions and Future Work

Conclusions

In this research, a systemic literature review was performed to analyze whether fall prediction had been performed at a great length previously. It was identified that fall prediction had not been done to a large degree and never while utilizing human pose estimation. This work sought to investigate whether human pose estimation had become effective enough to be used in a general sense, to aid in predicting falls. Along with this work, utilizing angle calculations and time-series forecasting, the question was posed if falls can be predicted ahead of time before they occur.

The output of this work shows that human pose estimation has become very generally effective to the point of enabling fall detection to a high degree. In addition, the output of human pose estimation can be combined with time series forecasting to predict future falls if only for a short degree as of the research of this thesis.

Future Work

While this thesis sought to explore the possibility and demonstration of fall prediction, there are many questions left unanswered and new questions have arisen. One large body of work left is exploring how best to improve the time series forecasting element. It is hypothesized, after performing this body of work, that a sequence-to-sequence translation model may work better than a time series forecasting model. This work is often more associated with neuro linguistics programming work and models that translate from one language to another. However, a cursory investigation while performing this work appeared to reveal that such a model may perform at a higher level and with greater accuracy.

In addition, while the angles that were calculated helped model performance, it is hypothesized that there may be odd angles and joint combinations for angle calculations that produce valuable information for a deep learning model. A body of work simply around creating all possible angle combinations and then pairing these calculations down to the best

ones may provide significant value. An extension of this would be exploring whether falls can be detected from angles alone without any key points. By calculating angles for all of the joints and detecting falls based on these angles, the problem can be moved more into the direction of a percentage risk of falling within a certain time period.

Lastly, any exploration into improving individual model performances would aid this process and future outcomes of similar work. While this work shows it is possible to predict falls, the precision is most likely hindered by small losses of precision and accuracy at each stage. These small defects in accuracy cascade further into the model with each stage. This is a hypothesis around the oddball result that was found whereby the best fall prediction model was a worse fall detection model. This model had no sliding window and thus was only evaluating a single row. When evaluating many rows, these small errors multiplied up to impact the accuracy of the model rather than providing the model with additional information.

References

[1] "Facts about falls," Aug 2021. [Online]. Available: https://www.cdc.gov/falls/facts. html

[2] "Falls." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/falls

[3] "Who global report on falls prevention in older age." [Online]. Available: https://www.who.int/publications/i/item/9789241563536

[4] B. Moreland, R. Kakara, and A. Henry, "Trends in nonfatal falls and fall-related injuries among adults aged 65 years — united states, 2012–2018," *MMWR. Morbidity and Mortality Weekly Report*, vol. 69, no. 27, p. 875–881, 2020.

[5] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez, and C. Peñafort-Asturiano, "Up-fall detection dataset: A multimodal approach," *Sensors*, vol. 19, no. 9, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/9/1988

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[7] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[8] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[9] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[10] V. Mayer-Schonberger and K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Boston: Houghton Mifflin Harcourt, 2013. [Online]. Available: http://www.amazon.com/books/dp/0544002695

[11] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[12] P. Cunningham, M. Cord, and S. J. Delany, *Supervised Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. [Online]. Available: https://doi.org/10.1007/978-3-540-75171-7_2

[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[14] Y. Tai, "A survey of regression algorithms and connections with deep learning," 2021. [Online]. Available: https://arxiv.org/abs/2104.12647

[15] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 334–340, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2314728817300715

[16] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, 04 1998.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[19] W. Wei, *Time Series Analysis: Univariate and Multivariate Methods*, ser. Time Series Analysis: Univariate and Multivariate Methods. Pearson Addison Wesley, 2006. [Online]. Available: https://books.google.com/books?id=aY0QAQAAIAAJ

[20] A. Toshev and C. Szegedy, “Deeppose: Human pose estimation via deep neural networks,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.214

[21] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, “Joint training of a convolutional network and a graphical model for human pose estimation,” 2014.

[22] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation,” 2016.

[23] K. Sun, B. Xiao, D. Liu, and J. Wang, “Deep high-resolution representation learning for human pose estimation,” 2019.

[24] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, “Multi-context attention for human pose estimation,” 2017.

[25] B. Artachos and A. Savakis, “Omnipose: A multi-scale framework for multi-person pose estimation,” 2021.

[26] B. Artacho and A. Savakis, “Unipose: Unified human pose estimation in single images and videos,” 2020.

[27] A. Núñez-Marcos, G. Azkune, and I. Arganda-Carreras, “Vision-based fall detection with convolutional neural networks,” *Wireless Communications and Mobile Computing*, vol. 2017, p. 1–16, 2017.

[28] Kong, Chen, Wang, Chen, Meng, and Tomiyama, “Robust self-adaptation fall-detection system based on camera height,” *Sensors*, vol. 19, no. 17, p. 3768, 2019.

[29] A. Iazzi, M. Rziza, and R. Oulad Haj Thami, “Fall detection system-based posturerecognition for indoor environments,” *Journal of Imaging*, vol. 7, no. 3, p. 42, 2021.

[30] S. Chhetri, A. Alsadoon, T. Al-Dala'in, P. W. Prasad, T. A. Rashid, and A. Maag, "Deep learning for vision-based fall detection system: Enhanced optical dynamic flow," *Computational Intelligence*, vol. 37, no. 1, p. 578–595, 2020.

[31] B.-H. Wang, J. Yu, K. Wang, X.-Y. Bao, and K.-M. Mao, "Fall detection based on dual-channel feature integration," *IEEE Access*, vol. 8, p. 103443–103453, 2020.

[32] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[33] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. delRío, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[34] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[36] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python,"*Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[38] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[39] G. Van Rossum, *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[40] B. S., T. Buitrago, P. Hanna, E. Sanielevici, S. Scibek, R., and N. N. A., "Bridges-2: A platform for rapidly-evolving and data intensive research. in practice and experience in advanced research computing (pp. 1-4)," 2021.

[41] T. J., C. T., D. M., F. I., G. K., G. A., H. V., L. S., L. D., Peterson, G. Roskies, R. S. J.R., and W.-D. N., "Xsede: Accelerating scientific discovery. computing in science engineering. 16(5):62-74." 2014.

[42] R. Mundani, J. Frisch, V. Varduhn, and E. Rank, "A sliding window technique for interactive high-performance computing scenarios," *CoRR*, vol. abs/1807.00092, 2018. [Online]. Available: http://arxiv.org/abs/1807.00092

[43] C. Sammut and G. I. Webb, Eds., *Mean Squared Error*. Boston, MA: Springer US, 2010, pp. 653–653. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_ 528