6-18-2021

The development of fully automated RULA assessment system based on Computer Vision

Gourav Kumar Nayak
University of Windsor

The development of fully automated RULA assessment system based on Computer Vision

By
Gourav Kumar Nayak

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of **Mechanical, Automotive and Materials Engineering**
in Partial Fulfillment of the Requirements for
the Degree of **Master of Applied Science**
at the University of Windsor

Windsor, Ontario, Canada

2021

The development of fully automated RULA assessment system based on Computer Vision

By

Gourav Kumar Nayak

APPROVED BY:

_____

R. Razavi-Far

Department of Electrical and Computer Engineering

_____

B. A. Schuelke-Leech

Department of Mechanical, Automotive and Materials Engineering

_____

E. Kim, Advisor

Department of Mechanical, Automotive and Materials Engineering

March 22, 2021

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

The purpose of this study was to develop an automated, RULA-based posture assessment system using a deep learning algorithm to estimate RULA scores, including scores for wrist posture, based on images of workplace postures. The proposed posture estimation system reported a mean absolute error (MAE) of 2.86 on the validation dataset obtained by randomly splitting 20% of the original training dataset before data augmentation. The results of the proposed system were compared with those of two experts' manual evaluation by computing the intraclass correlation coefficient (ICC), which yielded index values greater than 0.75, thereby confirming good agreement between manual raters and the proposed system. This system will reduce the time required for postural evaluation while producing highly reliable RULA scores that are consistent with those generated by manual approach. Thus, we expect that this study will aid ergonomic experts in conducting RULA-based surveys of occupational postures in workplace conditions.

Keyword: RULA, deep learning algorithm, musculoskeletal injuries, automated posture assessment system

Acknowledgements

At the very outset and foremost I would like to express my most sincere gratitude and thanks to my supervisor Dr. Eunsik Kim for kindly providing me the opportunity to work on this thesis project and for being a guiding light throughout the course of this thesis. His utmost patience, perseverance, resourcefulness and tireless supervision have made possible the completion of this project. It has been an absolute honor to have worked with him which has taught me a great deal about impeccable professional and ethical conduct as well invaluable knowledge about engineering concepts and principles.

I would also like to thank my thesis committee members Dr. B. A. Schuelke-Leech and Dr. Roozbeh Razavi-Far for their critical review and valuable advice.

Finally, I would like to thank the Department of Mechanical, Materials and Automotive Engineering for Graduate Assistantship (GA) opportunity, special thanks to the secretarial services of Ms. Angela Haskell and all faculty and staff members of the Department of Mechanical, Materials and Automotive Engineering and the Faculty of Graduate Studies for their help.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
ALLA	Agricultural Lower-Limb Assessment
AWS	Amazon Web Services
CDC	Centers for Disease Control and Prevention
CNN	Convolution Neural Network
COCO	Microsoft Common Objects in Context
DL	Deep Learning
DNN	Deep Neural Network
GCP	Google Cloud Platform
GPU	Graphical Processing Unit
ICC	Intra-Class Correlation Coefficient
IMU	Inertial Measurement Unit
LUBA	Loading on the Upper Body Assessment
MAE	Mean Absolute Error
NERPA	Novel Ergonomic Postural Assessment Method
NIOSH	The National Institute for Occupational Safety and Health
OWAS	Ovako Working Posture Analysis System
RCNN	Region based Convolution Neural Network
REBA	Rapid Entire Body Assessment
RL	Reinforcement Learning
RULA	Rapid Upper Limb Assessment
SL	Supervised Learning
STHMD	See-Through Head-Mounted Display
UL	Unsupervised Learning
WMSD	Work-related musculoskeletal disorder

Chapter 1 – Introduction

1.1 Background

Ergonomics is the study and practice of designing tasks and workplaces to fit the capabilities of workers without impacting the efficiency of both men and machines (Heinemann, 1974). Ergonomic designs are aimed to increase productivity in a workplace along with worker comfort and reduction of muscle fatigue. In 1997, the National Institute for Occupational Safety and Health (NIOSH) released a review of evidence that suggests relationships between work conditions and Work-related musculoskeletal disorders (WMSDs) (Bernard, 1997). WMSDs are injuries to the limbs of workers induced or aggravated by the working conditions in a workplace (Schneider et al., 2010). WMSDs occur when the physical ability of the worker is insufficient to meet the physical requirements of the tasks at the workplace. The working conditions that may lead to WMSDs include routine lifting of heavy objects, daily exposure to whole-body vibration, routine overhead work, work with the neck in a chronic flexion position, or performing repetitive forceful tasks (Bernard, 1997). WMSDs can affect body parts depending on the nature of tasks; for instance, tasks that involve using the upper body may cause pains in the upper arm, lower arm, wrists, neck, and shoulders, whereas tasks involving the lower body may affect legs, trunk, and feet (WMSDs, 2014). Carpal Tunnel Syndrome (Fig. 1.1), thoracic outlet syndrome (Fig. 1.2), and tendonitis (Fig. 1.3) are a few examples of WMSDs.

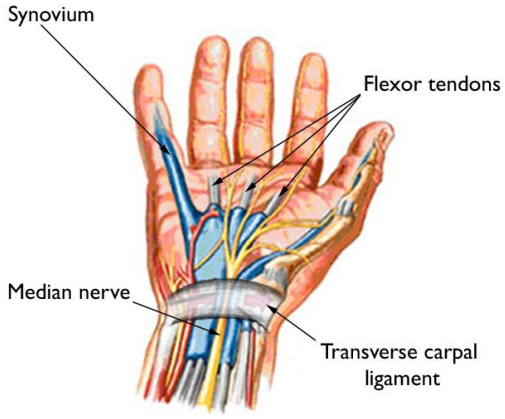

Figure 1.1: Carpal Tunnel syndrome (Image Source: prestige.com)

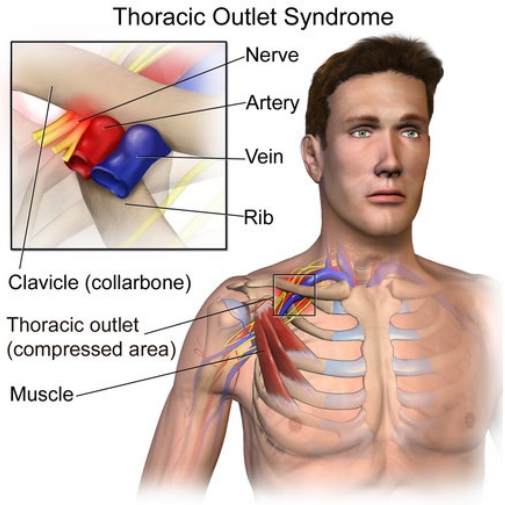

Figure 1.2: Thoracic Outlet syndrome. (Image Source: wikipedia.org)

Figure 1.3: A) Elbow tendonitis. Image Credits: stadiasportsmedicine.com B) Wrist Tendonitis. Image Credits: wataugaortho.com C) Knee Tendonitis. (Image Source: mayoclinic.org)

According to researchers from the National Reference Center for Rare Autoimmune Diseases at the University Hospitals of Strasbourg, WMSDs are ranked second worldwide in shortening people working years, following mental illness and substance abuse (Sebbag et al., 2019). WMSDs accounted for 4.1 million early deaths in the year 2015, an increase of 46 % since the year 2000 (Sebbag et al., 2019). WMSDs have contributed to almost 400,000 injuries, costing industries over $20 billion per year (Middlesworth, 2020).

According to the reports from the Centers for Disease Control and Prevention, the Worker Health Chartbook published under NIOSH, states that in 2001, WMSDs involved a median of eight days away from work compared to six days for all nonfatal injury and illness cases (Work-Related Musculoskeletal Disorders & Ergonomics). The Institute in Medicine estimates the economic burden of WMSDs between $45 and $54 billion annually as measured from compensation costs, lost wages, and lost productivity. As per the Liberty Mutual, the largest workers' compensation insurance provider in the United States,

overexertion injuries from lifting, pushing, pulling, holding, carrying, or throwing an object cost employer $13.4 billion every year (Work-Related Musculoskeletal Disorders & Ergonomics). Fig. 1.4 shows the rate of workers' compensation claims for WMSDs resulted lost-time per 10,000 employees, among the WRT NAICS subsectors with the highest rates in Ohio, United States between 2005–2009 (MMWR, 2013).

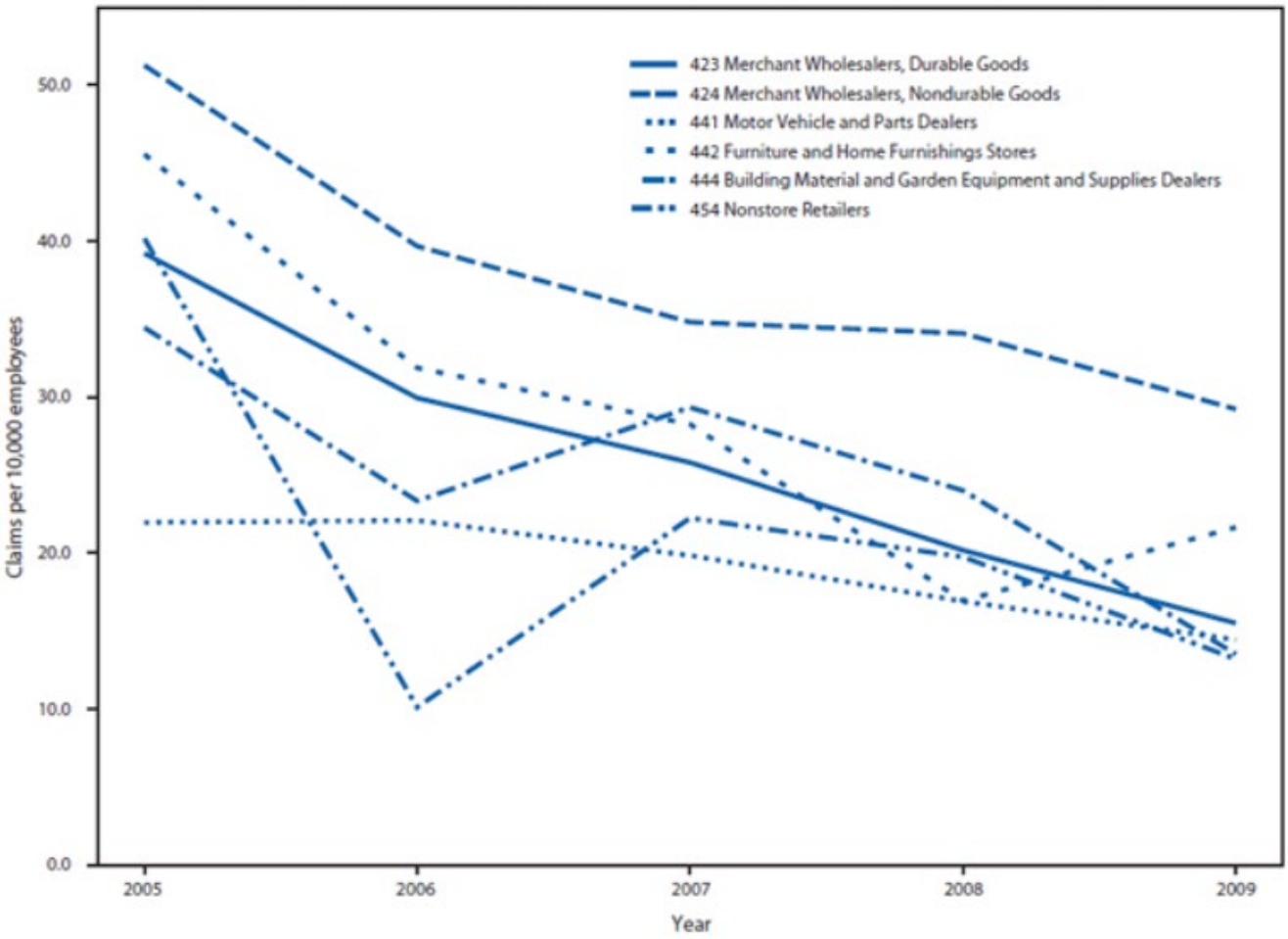

Figure 1.4: Number of time-loss claims for musculoskeletal disorders WRT NAICS subsectors in Ohio, United States.

The situation is not much different in the Canada. According to the 2016 reports from the Canada Public Health & Safety Association, WMSDs have contributed for almost 30% lost-time claims, causing approximately 2.3 million people repetitive strain injuries (Macpherson et al., 2018). A total of 1.2 million WMSDs claims were compensated for time-loss in the Canadian jurisdictions during 2004–2013 and resulted in time-loss equivalent to 239,345 years in the Canadian jurisdictions (Macpherson et al.,

2018). Fig. 1.5 shows that there were 1,194,393 WMSDs claims in Canada between the years 2004-2013 based on sex and age group (Macpherson et al., 2018).

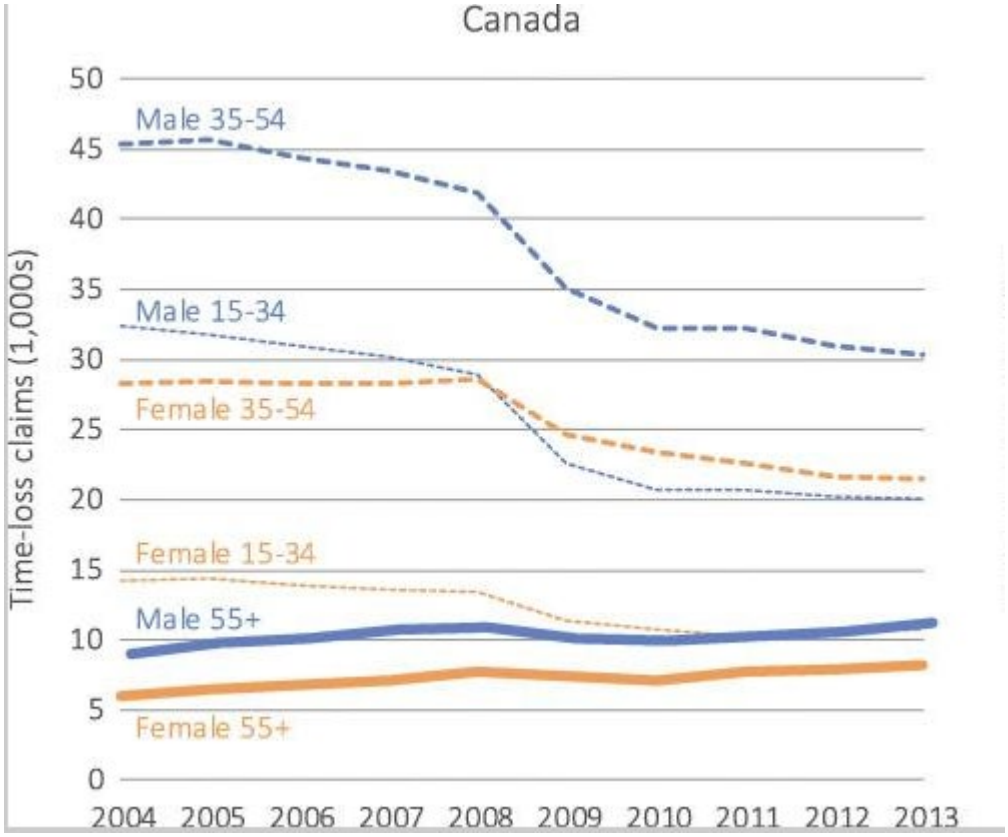

Figure 1.5: Number of time-loss claims for musculoskeletal disorders by sex and age group in Canada.

Previous studies have identified strong associations between WMSDs and awkward postures in the workplace (Anderson et al. 1997; Simoneau et al., 1996; Van Wely, 1970). Awkward working postures are the postures in which body parts deviate significantly from their neutral position while performing work activities (Yale Environmental Health and Safety, 2018). Fig. 1.6 shows examples of awkward working postures. Some of the examples of awkward working postures are working with hands above the head, elbows above the shoulder, working with neck or back bent to a high degree without any support, etc. Bending, pulling, lifting, twisting are examples of movements that are common in occupational workplaces but lead to WMSDs due to continual repetition and lack of recovery time

between these movements (WMSDs, 2014).

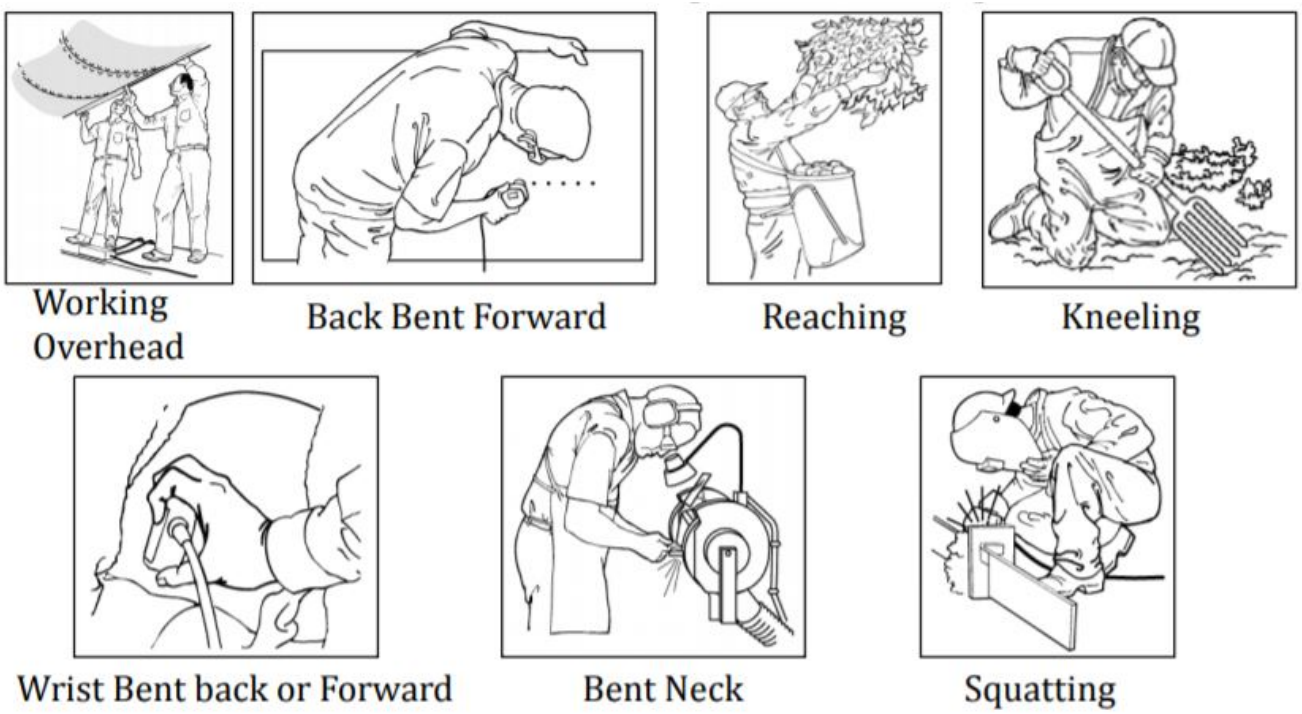

Figure 1.6: Awkward Working Posture (Image source: colby.edu)

There are various techniques to correct awkward working postures at any workplace, which involves redesigning of workplace, tools and equipment design, tasks redesign, etc. Ergonomists must identify awkward working postures in a workplace before taking measures to correct them. Thus, researchers have developed postural evaluation techniques to identify tasks that expose workers to WMSDs in the workplace. These techniques fall into three categories: self-reports, observational methods, and direct methods (David, 2005). Researchers use self-report methods to gather data that can contribute to workers' health via interviews or questionnaires from workers about their workplaces. These methods have the advantage that a large population can be surveyed in less time using a web-based questionnaire or video recordings of tasks. A major problem with self-report techniques is that subsequent analysis of data can be expensive and requires appropriate skills to interpret the findings. Also, the response to the survey questions are subjected to workers' perception of the difficulty of tasks which makes the analysis unreliable (Pope et al., 1998). Although the posture with high risks can be identified by self-report methods, the absolute measure of risk is not possible (Pope et al., 1998). As the name suggests, observational techniques require an ergonomic expert observe the workers performing the tasks, manually segment the relevant body parts and evaluate posture on different factors such as repetitive motion patterns, duration of work, and muscle force exertion (Andrews et al., 2012). The tools used in an observational method for postural assessment measures the exposure to risk factors on different scales

and for different body segments; for instance, Ovako Working Posture Analysis System (OWAS) does not include neck, elbows, and wrist in the evaluation, whereas Rapid Upper Limb Assessment (RULA) does. Similarly, RULA has four risk levels that represents the level of WMSD risk of the posture, while Rapid Entire Body Assessment (REBA) has five risk levels. These techniques suffer from inter-and intra-observer variability when choosing between different categories of risk exposure and are only suitable for static postures (Beek and Frings, 1998). Also, the evaluation of posture using observational methods requires the investigator to be trained on conducting the survey based on the selected tool, which makes the technique expensive. Direct methods is the third approach to evaluate postures in a workplace and it utilizes advanced technologies such as wearable devices (Peppoloni et al., 2015) and Kinect-based systems (Plantard et al., 2017) for online assessment of WMSD risks. Direct measurement techniques are used to gather huge amount of data quickly and eliminate the need to manually segment the body parts. The results of the direct measurement techniques can be used to evaluate body postures exposure to risk factors without much human interaction as compared to observation methods. But the attachment of sensors on worker's bodies may result in discomfort to workers and is not always possible; for instance, in a high-temperature environment or tasks that require protective clothing on workers. Also, the technique requires purchasing expensive equipment for evaluation of posture, which involves high initial cost in addition to cost of training the investigators for conducting the investigation. The direct measurement techniques are discussed further in detail in section 2.3.

1.2 Problem Statement

The observation-based posture assessment method requires an investigator to record videos of workers performing the tasks at the workplace then, manually segment the body parts in each frame and evaluate the posture using a worksheet-based assessment system which provides a rating based on the movement of body parts, task frequency, load lifted, etc. Fazi et al. (2019) conducted a risk assessment study at an automotive manufacturing company in three phases: Phase 1 was the identification stage which involved interviewing the complete production line in order to study the working condition of the plant and identify the most critical risk workplace. Phase 2 was called the empirical stage, in which the investigator collected anthropometric data of the workers and recorded them while performing tasks. The recordings took 2 hours for only nine operators working at a welding spot-gun assembly line. The recordings were reviewed, and three postures were selected for phase 3 based on the repetition of postures, time range, and awkward conditions. The postures selected were evaluated using RULA to identify the risk associated with the postures in phase 3, called the Analytic stage. The study makes it evident that

selecting posture for evaluation using the observation-based technique is subjective rather than being objective. Also, the data was collected for only a single task and a limited number of workers, which is an ideal condition. In practice, the sample size is large which takes a huge amount of time in data collection and reviewing postures via video recordings. The postures selected in phase 2 are evaluated using a worksheet-based postures evaluation tool which provides a rating to individual body parts, which leads to an entire body posture evaluation. This process needs to be repeated for every posture manually, which is cumbersome and prone to human error. Additionally, the investigator needs to be trained on data analysis and posture assessment tools, which incurs training and labor cost. The proposed method seeks to address these issues by developing a computer vision-based system that requires investigators with even no prior training or knowledge of RULA to record postures using video-cameras in a real workplace, and postural evaluation results will be provided by the system in real-time.

1.3 Statement of Purpose

The purpose of this study is to develop a fully automated system based on computer vision to evaluate full body postures in a workplace based on the RULA Posture assessment tool. More specifically, this study is about developing a computer vision model that can predict the location of human body joint coordinates from the images of workers in real workplace conditions performing common occupational tasks like pulling, lifting, pushing, machining, etc. The proposed algorithm computes the angle between different body segments in the Frontal and Sagittal plane and evaluates relevant body postural scores, including wrist scores that all the previous studies which utilizes machine learning technique failed to compute. These scores are translated to RULA Grand scores which denotes the risk level associated with work posture. The reliability of the proposed algorithm is validated statistically with the manual evaluation from two ergonomic experts by computing the Intra-Class Correlation Coefficient (ICC) and independent sample t-test for occupational postures recorded under different work conditions and performing a variety of tasks. This method reduces the time required for RULA evaluation by eliminating the need for investigators to spend time sampling and evaluating posture from video camera recordings of workplace tasks. This study aims to determine whether it is possible to develop an automated system based on a deep learning algorithm that will reduce evaluation time and produce RULA score results that are sufficiently similar to those generated by manual evaluators in observational postural assessment.

1.4 Hypothesis

1. The proposed posture evaluation method should yield a high agreement to the ratings provided by manual posture assessment for occupational postures in a real workplace.

   Intra-Class Correlation Coefficient (ICC) serves as a reliability index that reflects both degrees of correlation and agreement between the results of two evaluation methods, comparing specifically the ratings from 2 ergonomics experts and those from the proposed machine learning algorithm in this study. This index is used when subjects are chosen at random and raters are fixed. An ICC index that lies between 0.75-0.90 is interpreted as a very good agreement between the raters, whereas an ICC index greater than 0.90 indicates excellent reliability (Koo and Li, 2016).

2. There should be no effect of the technique used for the assessment of postures on the RULA posture evaluation results.

   An independent sample t-test test was used to evaluate the impact of the technique used for posture assessment on the RULA-based posture evaluation. An independent sample t-test was conducted between 2 independent groups (Manual Evaluations and Proposed method Evaluations) to determine whether there is any statistically significant difference in the evaluations obtained. A p-value greater than 0.05 for a 95% Confidence Interval indicates no significant effect of the evaluation technique on the RULA posture evaluation.

Chapter 2 – Literature Review

2.1 Observational Postural Evaluation Techniques

This section discusses observation-based posture evaluation tools that are used by ergonomic experts in the industry to assess WMSDs risks associated with postures in a workplace. In an observation-based approach, an investigator analyses a work-posture in real-time or via recordings using video-cameras for an entire work-cycle manually segments the relevant body parts and assigns the postural risk score based on the tool worksheet. The worksheet includes scoring criteria for a range of body angles which varies from one tool to another. Various postural evaluation tools like the Ovako Working Posture Analysis System (OWAS) (Karhu et al., 1977), the Novel Ergonomic Postural Assessment Method (NERPA) (Sanchez-Lite et al., 2013), the Rapid Upper Limb Assessment (RULA) (McAtamney and Corlett, 1993) and the Rapid Entire Body Assessment (REBA) (Hignett and McAtamney, 2000) have been developed to evaluate postural risks associated with workplace tasks. Each technique has its own posture classification scheme, which may result in the assignment of different postural load scores for a given posture, depending upon the method used. These tools are used for initial screenings of postures that can lead to WMSDs, and that may require further analysis with more comprehensive tools.

2.1.1 The OWAS Method

The OWAS was developed in the year 1973 in Finland in a steel industry named OVAKO OY, which is a leading producer of steel bars and profiles in Europe (Karhu et al., 1977). OWAS is an observational method for analyzing work postures based on frequency or time spent in each posture in a work-cycle. OWAS systematically classifies a work-posture into one of the 252 (= 4x3x7x3) possible combinations based on the back (4 categories), upper limb (3 categories) and lower limb posture (7 categories), and weight of load or amount of force used (3 categories) as shown in Fig. 2.1.

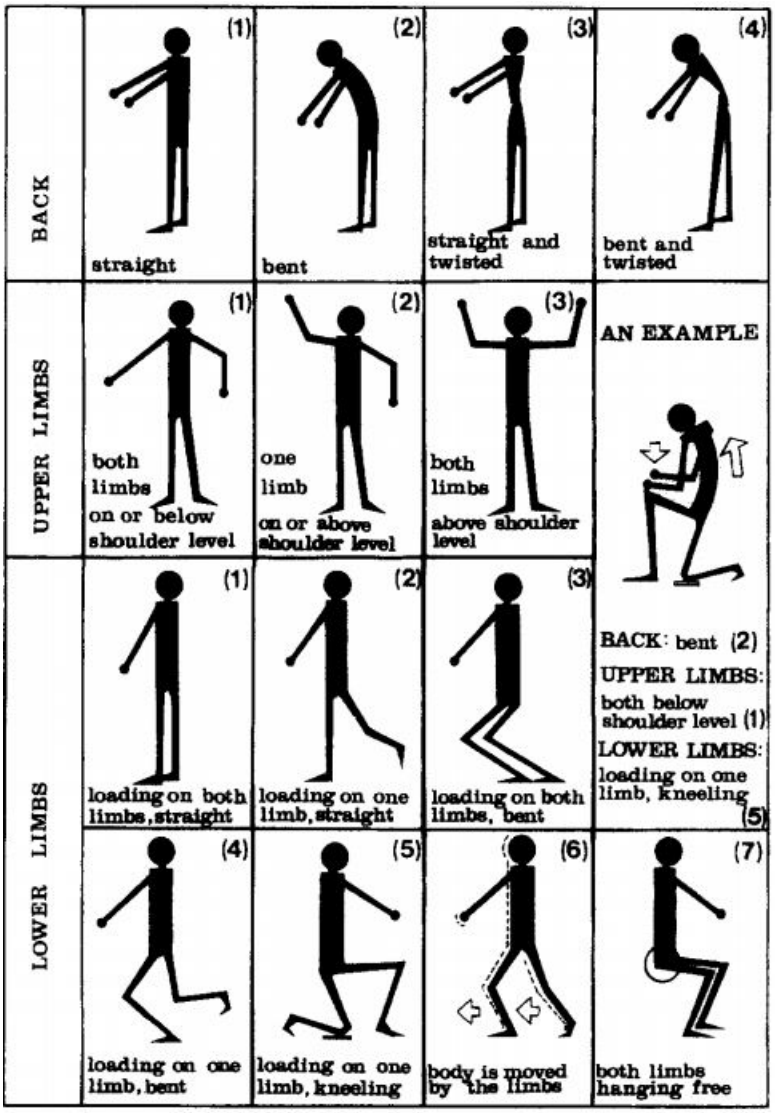

Figure 2.1: OWAS body posture classification codes. (Image source: Karhu et al., 1977)

The resulting work posture is further classified into four action categories (Table 2.1) that indicate the risk of injury associated and hence, the priority to take corrective actions (Lee and Han, 2013).

Table 2.1: OWAS Action category

Action Category (AC)	Action
AC-1	Normal postures don't require special attention.
AC-2	Posture has some WMSDs risks, corrective action required in the near future.

AC-3	Posture has a harmful effect on the musculoskeletal system; corrective actions need to be taken as soon as possible.
AC-4	Posture needs immediate corrective action, must be the highest priority.

OWAS has been used for posture assessment in many industries such as Construction site (Lee and Han, 2013), Forestry (Justavino et al., 2015), Nursing (Engels et al., 1994; Hignett, 1994), etc. due to ease in implementing and availability of wide research in various occupations (Takala et al., 2010). Despite these advantages, OWAS is quite ineffective for full-body posture evaluation as it does not include neck, trunk, elbows, and wrists in its evaluation. Also, OWAS does not consider repetition or duration of the sequential postures (Takala et al., 2010).

2.1.2 The REBA Method

The REBA method is a postural analysis system sensitive to WMSDs risks developed to analyze and evaluate working postures in occupational workplaces, mainly healthcare and other service industries (Hignett and McAtamney, 2000). REBA provides a scoring system for the upper and lower body based on the degree of movement of individual body parts along with force or load exerted on the body in work. REBA requires an ergonomic expert to select the working posture based on the difficulty of the task (information gathered from interviewing workers), the most common posture in a work cycle, and the posture where the highest amount of load occurs. Then, the selected posture is evaluated by manually segmenting the body parts and calculating the angle of the trunk, upper arm, lower arm, neck, and leg for assigning individual body posture scores as per the REBA guidelines shown in Fig. 2.2. The angles of individual body parts can be calculated using a digital goniometer while performing tasks, or the tasks can be recorded using video cameras, and angles can be calculated with online protractor tools. The force or load scores can be between 0-2 based on the load of the weights and if there is rapid shock involved in the task. REBA also takes into account the coupling score, which ranges from 0-3 (lower is the better) based on the quality of grip; for instance, coupling score is 0 when there is a well-fitted handle while it is three if there is no handles and unsafe reach involved. The scores for upper arm, lower arm, and wrist are used to calculate upper body posture score as shown in Table B, and neck, trunk, and leg scores are used for lower body posture score given in Table A (Tables are part of worksheet given in Fig. 2.2). The upper and lower body posture scores with Force/Load score and coupling score give a resultant grand

score between 1-15 as per Table C. The grand score is increased depending on the activity score by one if the body parts are held longer than one minute or repeated more than four times per minute and by two for repeated small range actions(more than 4x per minute) and plus three for action that causes large rapid changes in postures. The resultant grand score is categorized into five action levels that indicate the risk of WMSDs associated with the posture and order of the priority of ergonomic intervention required in the task given in Table 2.2.

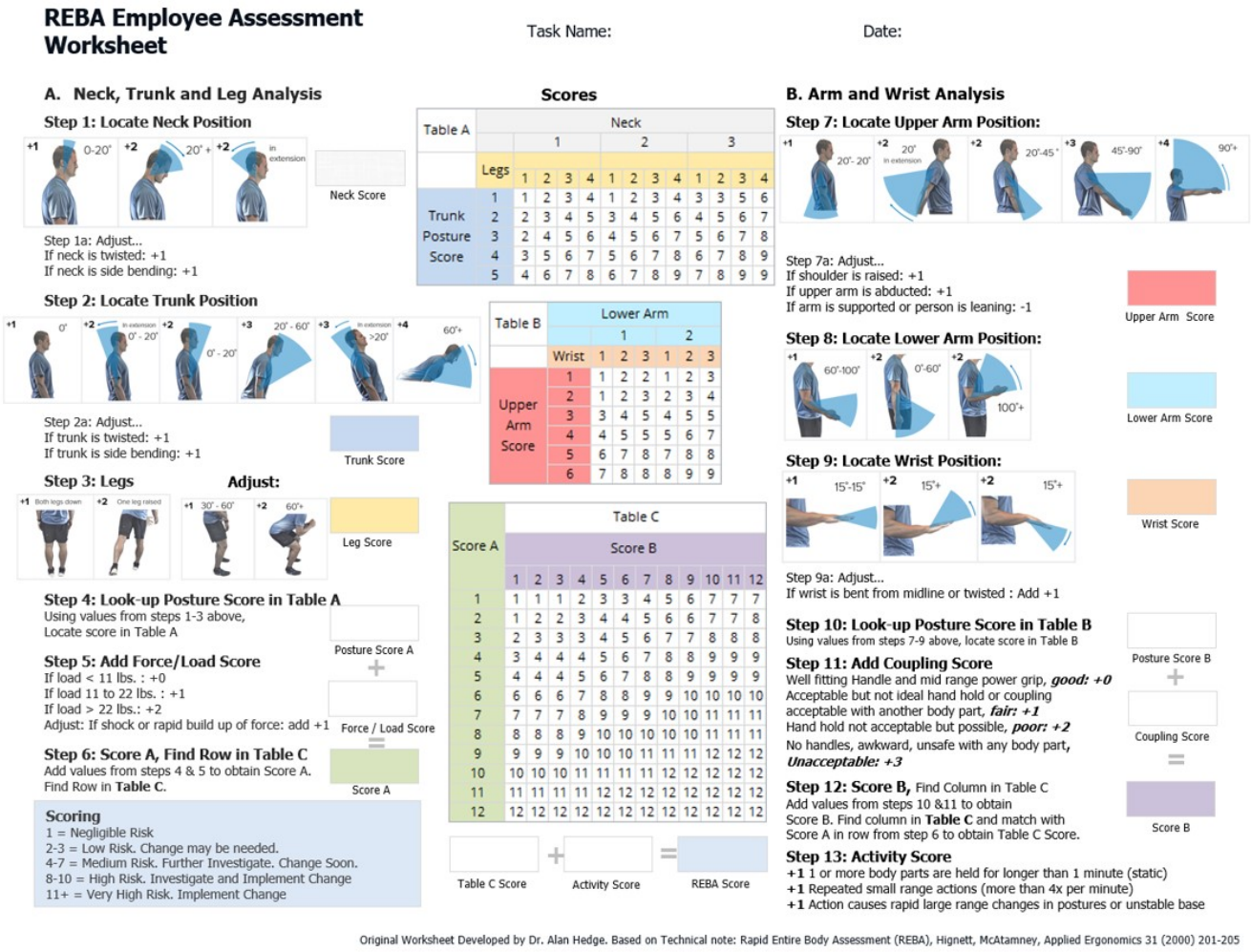

Figure 2.2: REBA Assessment Worksheet. (Image Source: ErgoPlus.com)

REBA is easy to be implemented in industries and doesn't require any technical equipment. In addition to this, REBA provides an evaluation of full-body posture, including wrists, neck, and trunk, which were unavailable in OWAS. REBA posture assessment tool has been used in many industries for postural evaluation such as Agriculture (Das and Gangopadhyay, 2015; Das et al., 2012; Das et al., 2013), manufacturing (Abaraogu et al., 2016; Yanes et al., 2012), Forestry (Gallo and Mazzetto, 2013;

Houshyar and Kim, 2018; Enez and Nalbantoğlu, 2019) and other activities (Asadi et al., 2019; Diego-Mas et al., 2017; Dumas et al., 2014; Das, 2015). Like any other tool, REBA has its limitations, such as REBA doesn't consider the duration of tasks or frequency of postures. Also, REBA only allows the analysis of individual postures, and it is not possible to analyze a set or sequence of postures.

Table 2.2: REBA Action Scores

Score	Level of WMSDs Risk
2	Negligible risk, no action required
2-3	Low risk change may be needed
4-7	Medium risk, further investigation, change soon
8-10	High risk, investigate and implement change
11+	Very high risk, implement change

2.1.3 The RULA Method

The RULA is an observational survey method developed to provide an assessment of the upper body posture in an occupational workplace task reporting cases of WMSDs risks (McAtamney and Corlett, 1993). The working posture is recorded by an investigator with the help of a digital camera or a video recorder. The side of the posture to be investigated is decided based on external load factors, which include a number of movements, static muscle work, and force (McAtamney and Corlett, 1993). These details are identified during a workplace investigation. Both sides of the posture are recorded in case the investigation results are not enough to make a decision. RULA divides the full-body posture for each side into two segments - Group A and Group B. Group A includes the upper arm, lower arm, and wrist, while Group B includes the neck, trunk, and legs. RULA comes up with a scoring system that provides the extent of postural loading on the musculoskeletal system. An investigator studies the range of movements for each body part from the recordings and assigns a score formulated in the RULA study as shown in Fig. 2.3. Group A and Group B postural scores are adjusted in accordance with Muscle Use or Load exerted on the body. The muscle use score values one if the posture is mainly static (held >1 minute) or repeated (occurs at least four times per minute). The Force/Load score may vary between 0-3 depending on the load and posture occurrence; for instance, the load score will be 0 if the amount of load carried is < 4.4lbs. And posture is static, whereas the load score will be three if the load is more

than 22lbs. And posture is repeated, or tasks cause shocks such as hammer use.

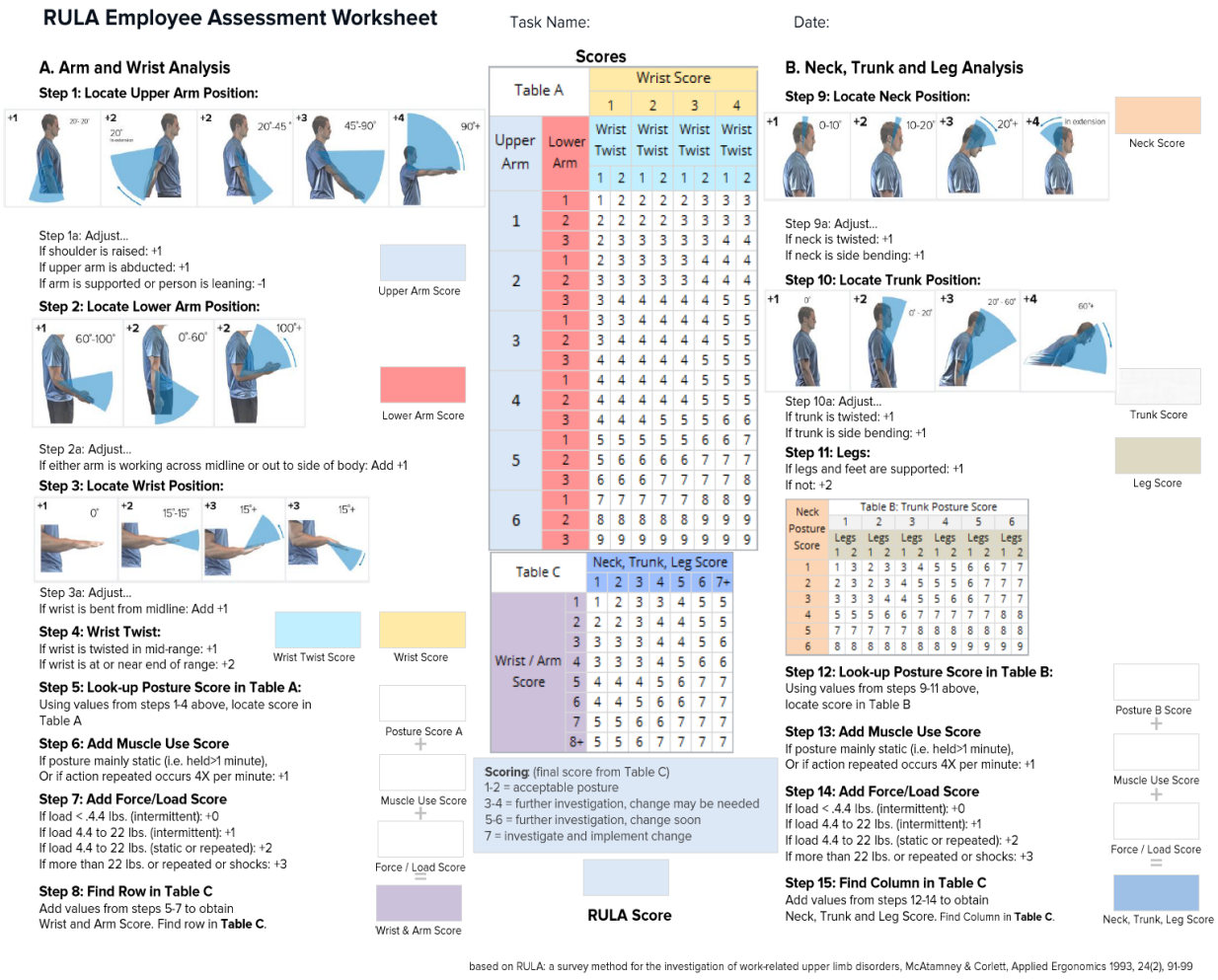

Figure 2.3: RULA Posture Assessment Worksheet. (Image Source: ergoplus.com)

The scores from Group A and Group B combine to form a grand score. The grand score is decoded into four action levels which indicate the level of intervention required to reduce the risks of injury due to physical loading on the worker (McAtamney and Corlett, 1993).

Table 2.3: RULA Action Scores

Grand Score	Level of WMSDs Risk
1-2	Acceptable working posture if not maintained or repeated for long periods
3-4	Further investigation is needed; posture change may be required
5-6	Investigate and implement posture changes soon to avoid further exposure

	to WMSDs risk
7+	Requires immediate attention and changes in posture

The RULA survey tool has proven useful in postural assessments of such occupational fields and setting as supermarkets (Ryan, 1989), agriculture (Tuure, 1992), ship maintenance (Joode et al., 1997), soft drink distribution (Wright and Haslam, 1999), metalworking (Gonzalez et al., 2003), transport driving (Massaccesi et al., 2003), carpet mending (Choobineh et al., 2004), etc. RULA needs no special equipment to conduct the investigation at the workplace, which makes it easy for investigators to use. Furthermore, RULA allows for quick assessment of the upper body, making it popular and reliable in the industry (Kee, 2020; Kong et al., 2018). RULA considers the muscular effort which is associated with working postures and force exerted on the body while performing repetitive or static work, and which may contribute to muscle fatigue which is one of the main contributors of WMSDs (WMSDs-Risk Factors, 2014).

RULA, like any other posture assessment tool, is subjected to few weaknesses. RULA requires to select postures based on the investigation conducted in the workplace prior to the survey via questionnaire or personal interviews, which is time-consuming. Also, the response to survey questions are based on workers' perception of pain, which affects the result of the survey. The postures that are investigated using RULA are those that reported discomfort or pain, or occur frequently, or experiences heavy load due to the nature of the task. This method of sampling postures for investigation is subjective rather than objective which may result in an overlook of bad postures. RULA is a manual survey method that requires the investigators to be trained, which makes it expensive and prone to human errors. RULA posture evaluations are inconsistent, and inter-rater agreement is low in case of complex postures where the investigator needs to make judgment due to occlusion of body posture from the workplace (Takala et al., 2010).

Kee (2020) conducted a study to compare the evaluation of OWAS, RULA, and REBA postural assessment techniques based on postural loads at each action category level. The ergonomic investigator evaluated 301 postures sampled from various manufacturing industries, including iron and steel (68 postures), electronics (46 postures), automotive (44 postures), and chemical industries (66 postures), and the service industry of a general hospital (77 postures) using OWAS, RULA, and REBA. The author reported that OWAS failed to correctly identify postures in the iron and steel industry with high

biomechanical backloading as compared to RULA, even though OWAS was developed to assess postures in the steel industry (Kee, 2020). Also, REBA assessed all 77 postures in the general hospital with the same action level 2 and underestimated postural loads for these postures, compared with RULA (Kee, 2020). These results showed that OWAS and REBA generally underestimated postural loads for the analyzed postures as compared to RULA, irrespective of industry and work type (Kee, 2020). Similarly, Yazdanirad et al. (2018) conducted a study on 210 workers from three different industries, including pharmaceutical, automotive, and assembly in the Isfahan province, to compare RULA, Loading on the upper body assessment (LUBA), and new ergonomic posture assessment (NERPA) method and effectiveness of these ergonomic risk posture assessment methods on upper extremity musculoskeletal disorders. The postures were evaluated using the three posture assessment techniques and Nordic questionnaire to determine the level of WMSDs risks. The results were compared statistically using the Wilcoxon test, which showed that RULA was the best method for assessing musculoskeletal disorders among the three methods (Yazdanirad et al., 2018). These studies have inspired ergonomic analysts to use the RULA posture evaluation tool for the assessment of postures in occupational workplaces.

2.2 Deep Learning

Computer vision is an interdisciplinary scientific field that focuses on replicating human vision and perception of objects in images and videos using computer systems and use the gathered data to build autonomous systems (Huang, 1996). Larry Roberts, the father of Computer Vision, discussed the possibilities of extracting 3D geometrical information from 2D perspective views of blocks in his Ph.D. thesis at MIT (polyhedra), which led to a revolution in the field of computer vision (Aloimonos, 1990). Traditionally, the main application of Computer vision was to extract features such as color detection, edge detection, corner detection from the images using algorithms such as SIFT (Scale-Invariant Feature Transform) (Lowe, 2004), SURF (Speeded-Up Robust Features), or BRIEF (Binary Robust Independent Elementary Features) (Calonder et al., 2010). The problem with computer vision-based approach is that the features are required to be selected for feature extraction, which gets complex when the number of features is large. Deep Learning (DL) which is a subset of machine learning, has been proven effective in the field of feature engineering (Mahony et al., 2019). The main difference between DL and traditional computer vision techniques is that there is no requirement to define features for feature engineering as

the DL-based algorithm develops a model complex enough to predict output features from the input features (Fig. 2.4).

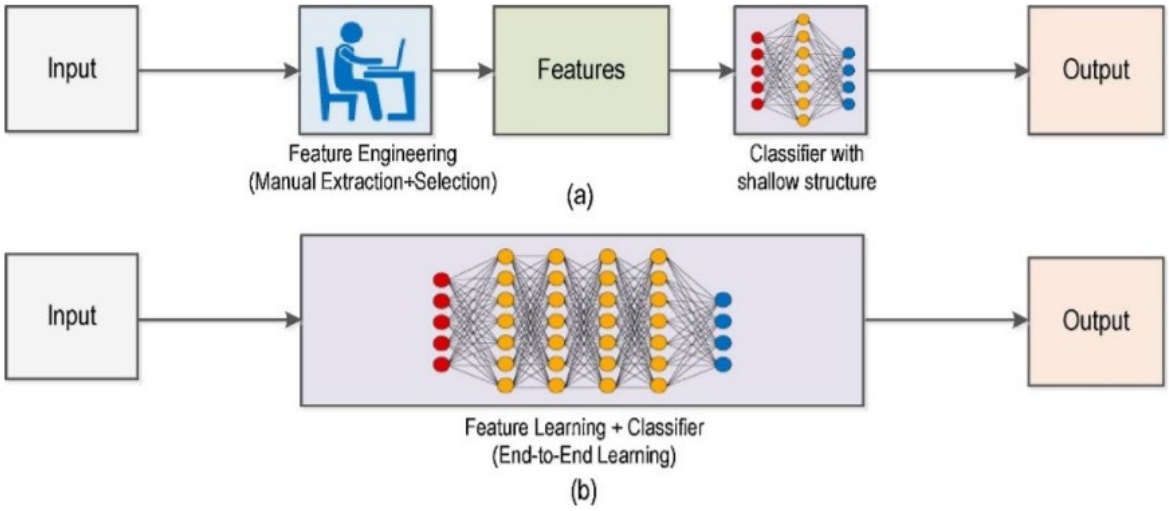

Figure 2.4:Traditional Computer vision workflow vs. Deep Learning workflow. (Image Source: Deep Learning for smart Manufacturing: Methods and applications)

DL is based on Artificial Neural Networks (ANNs), which is a computing paradigm inspired by the human brain. ANNs, like a human brain, is composed of computing cells called neurons that interact with each other to make a decision (Mahony et al., 2016). The problem with the application of DL is that it requires a huge amount of labeled data to train the model and high computing power to process these data. But the recent rise in availability of public datasets and cloud-based resources has increased the popularity of the computer vision-based deep learning approach. Also, the pipeline to train ANN does not require high programming skills due to the availability of popular machine learning libraries such as Keras and Pytorch, which has made the training process easier than ever.

2.2.1 ANN

An ANN is consist of computational units called neurons (McCulloch and Pitts, 1943) that are connected to each other in two or more layers to form a network architecture. Each neuron in a layer shares weights and outputs from the previous layer to generate output for the next layer. An ANN architecture has two common layers – Input and Output. The input layer accepts the input data, and the output layer generates the output of the network. The third layer is a hidden layer which are intermediate layers in between the input layer and output layer. A hidden layer consists of neurons that receive input from previous layers along with connection weights and passes it to the next layer after transforming the output with the help

of an activation function (Gallo, 2015). The computation power and complexity of an ANN increase with the number of neurons and the number of hidden layers.

A machine learning algorithm aims to extract the relation (or pattern) between input data and output data and encode that relation into parameters (weights) of the network (Alpaydın, 2014). The fundamental objective of the Machine Learning domain is to formulate algorithms that are capable of learning without much human assistance and intervention. There are three types of machine learning: Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL). SL is a machine learning paradigm for acquiring the input-output relationship information of a system based on a given set of paired input-output training samples, also called labeled training data (Liu and Wu, 2012). SL is further subdivided into two parts based on supervised data: classification and regression. If the algorithm predicts output as a finite set of discrete values that represents class labels of the input, the problem is considered as a classification problem, whereas if the output accepts continuous values, it leads to a regression problem. For instance, a problem for predicting cats or dogs in an image requires the algorithm to label cats as 0 and dogs as 1 or vice versa. The SL algorithm is trained with sample pair of inputs and outputs until a desirable accuracy is achieved. The supervised model classifies the new image as the probability of occurrence of the label. The same use case becomes a regression problem when the task requires to predict location of cats and dogs in the images by predicting the boundary coordinates. UL is a type of machine learning that seeks previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision (Siadati, 2018). In contrast to SL, UL recognizes the relationship in the data without the output attribute, and thus, all the features are treated as input. UL techniques are mainly used for data clustering, that is, to identify inherent groupings within the unlabeled data and subsequently assign a label to each data value (Dougherty et al., 1995). RL is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences (Bhatt, 2018). Though both SL and Rl use mapping between input and output, unlike SL, where feedback provided to the agent is a correct set of actions for performing a task, RL uses rewards and punishment for positive and negative behavior. As compared to UL, RL is different in terms of goals. While the goal in UL is to find similarities and differences between data points; in RL, the goal is to find a suitable action model that would maximize the total cumulative reward of the agent (Bhatt, 2018).

2.2.2 Research Domain of CNN

Convolution Neural Network (CNN) is a kind of ANN that has one or more layers of convolution units which reduces computational complexity and ensure translational invariance (Malmgren-Hansen et al., 2016) in terms of image recognition. CNNs are beneficial because they reduce the number of learnable parameters, which reduces the chances of overfitting as the model would be less complex than a fully connected network. CNNs are trainable multistage architectures with each stage consists of convolution layer, pooling layer, activation layer, and Fully connected layer. With the advent of low-cost Graphical Processing Units (GPUs) and cloud-based resources such as Amazon Web Services (AWS), Google Cloud Platform (GCP), etc., it has become possible to train deeper and complex CNNs. This has led to a huge rise in automating tasks in industrial applications. Some of these tasks are mentioned below:

a) Face Recognition: CNN-based models have been used for detecting the location of faces in an image and identifying them. Face recognition is a challenging problem because of the varying facial color, facial expression, occlusion, and illumination in the image. CNN is proven to be very accurate in solving this problem. Yang et al. (2018) has proposed facial recognition on uncropped face images using a deep convolution neural network. The CNN model was trained on CelebA (Liu et al., 2015) and AFLW (Köstinger et al., 2011) dataset. The Faceness-Net model pipeline is composed of two stages. The first stage of Faceness-Net applied a cascade of attribute networks to generate response maps of different facial parts such as hair, nose, eye, mouth, and beard that generate candidate windows (Yang et al., 2018). The second stage uses a multi-tasking CNN that refines the predicted candidate boxes from the first stage and performs face classification and bounding box regression (Yang et al., 2018). The proposed CNN architecture achieves promising performance on face detection benchmarks, including FDDB, PASCAL Faces, AFW, and the challenging WIDER FACE dataset. Facial Recognition tasks are useful in Biometric applications such as unlocking smartphones, identify people on social media platforms, criminal identification, etc.
b) Object Detection: It is the task of segmenting the objects such as a building, cars, houses, person, etc., from the background. An object detection model must be able to determine the location of the objects in the image (object localization) and predict the category of the object in the image or video (object classification). Redmon et al. (2016) have discussed the YOLO Object detection technique that can classify up to 20 objects and predict bounding boxes in natural images in the real-time. The YOLO network architecture has 24 convolution layers followed by two fully connected layers. The model has been pre-trained on ImageNet dataset (Russakovsky et al., 2015) at half the resolution

(224x224) and then doubled (448x448) for training on labeled images from PASCAL VOC 2007 and 2012 dataset for 135 epochs, batch size of 64, a momentum of 0.9 and a decay of 0.0005 (Redmon et al., 2016). The YOLO model processes images at 45 frames per second, which is faster than state-of-the-art objects detection models such as RCNN (Ren et al., 2017) and DPM. Some of the popular applications of real-time object detection algorithms are object detection using UAV (Kamate and Yilmazer, 2015), Handling of objects using Robotic arms (Chen et al., 2014), autonomous driving (Azhar, 2014), etc.

c) Text Classification: CNNs are used to extract features from unstructured textual data and convert it to structured data that can be assigned tags or categories based on the content using popular data mining techniques (Hu et al., 2018). Lewis (2016) presented a method combining CNN with the SVM technique to develop a deep learning model called SVMCNN that uses CNN to extract features of short texts and then uses SVM classifier for classification. The SVMCNN model is trained on sentiment polarity dataset (contains more than 5,000 movie-review data) (Movie Review Data) and Twitter dataset (contains 3,169 comments from the Twitter social platform) (Unstructured Text) labeled as positive or negative. The results show that the SVMCNN has good performance in unstructured text classification with a high Precision rate, Recall rate, and F1-measure (Lewis, 2016). Some of the popular applications of text classification methods in real life are spam filtering, product recommendation systems, etc.

2.2.3 CNN based Human Pose Estimation

Human Pose Estimation is the task of identifying the label and location of human body joints such as shoulder, elbow, knee, trunk, etc., in the image. The difficulty in predicting the body joints occur due to occlusion from clothing, surrounding, etc., poor lighting conditions and strong articulations. The Traditional approach to pose estimation uses a mixture-of-parts model to represent complex joint relationships (Computer Vision). The deformable model includes both a coarse global template that covers the entire human body and a higher resolution part template for smaller body joints (Unstructured Text). These templates represent Histogram of Gradient (HOG) features that are matched in an image to detect individual body parts connected to each other and arranged spatially in a different orientation. The limitation of this approach is the inability to detect a person in a crowded environment and with less visible body parts. Human Pose Estimation has significantly progressed with the advancement of Convolution Neural Networks (CNN) and popular keypoint datasets such as Microsoft Common Objects

In Context (COCO) (Lin. et al., 2014), MPII Human Pose Dataset (Andriluka et al., 2015) and Human 3.6M (Ionescu et al., 2014). Human pose estimation is classified as 2D Pose estimation and 3D pose estimation based on estimation of 2D pose coordinates (x, y) or 3D pose coordinates (x, y, z), respectively, for each joint from an image. The 3D pose estimation techniques uses the results from 2D pose estimation to predict 3D coordinates of human key joint by concatenating depth features obtained from the image (Sapp and Taskar, 2013). Toshev et al. (2013) were the first to propose a method for human pose estimation based on Deep Neural Networks (DNNs). The paper proposed a seven-layer AlexNet based neural network to directly regress the 2D location of body joints in a color image. The paper also proposed a cascade of neural networks to refine the prediction for a single body joint which enhanced the performance of the neural network (Toshev and Szegedy, 2013). The regression-based DNN has been proven very effective in predicting body joints that are heavily occluded and work well in a crowded environment as well. Another CNN-based approach was to use an iterative corrective mechanism for estimating the full-body posture in an image. Yang and Ramanan (2013) discussed in their study a self-correcting model that progressively changes an initial solution in every iteration by feeding back error predictions called Iterative Error Feedback (IEF). The CNN accepts an RGB image of size 224x224 and predicts 2D coordinates of 17 human keypoints. The model was tested on MPII Human Pose Dataset (Andriluka et al., 2015), which features significant scale variation, occlusion, and multiple people interacting, and Leeds Sports Pose dataset (LSP) (Carreira et al., 2016), which features complex poses of people in sports and shown excellent performance. One of the novel CNN architecture for Human Pose estimation that achieved state-of-the-art results on MPII Human Pose Dataset (Andriluka et al., 2015) and FLIC dataset (Newell et al., 2016) is in the form of a stacked hourglass network that consists of steps of pooling and upsampling layers stacked together to make a final set of predictions (Johnson and Everingham, 2010). The hourglass architecture is a simple, minimal design that has the capacity to capture multi-scale resolutions and bring them together to output pixel-wise predictions. The network uses a single pipeline with skip layers to preserve spatial information at each resolution and passes it along for upsampling further down the hourglass. The network uses intermediate supervision; that is, the predictions at each hourglass in the stack are supervised, and the feedback is used by the network to make final predictions.

The CNN architectures handle diverse and challenging sets of postures with occlusions from the surrounding environment and under different lighting conditions. The advantage of using CNN architecture for human pose estimation is translational and color invariance. The CNN architectures trained using a dataset with images of a person in real life can be modeled to handle the noise, which

makes it viable for practical application. Posture estimation has a wide range of applications such as activity recognition, motion capture, augmented reality, posture estimation, etc.

2.3 Advanced techniques for Posture Estimation

Posture assessment using objective measurements has been very popular because it eliminates the need for experts to manually segment body parts and evaluate movements and because technological advancements in the field of computing have made it easier than ever to access these tools (Lowe et al., 2019). The aid from objective measurement techniques may increase the speed and accuracy of posture assessment. Earlier attempts to achieve this were based on wearable devices (Peppoloni et al., 2015; Abobakr et al., 2017; Vignais et al., 2013; Li et al., 2014; Yan et al., 2017), Kinect based systems (Plantard et al., 2017) and machine learning algorithms (Li et al., 2020; Sasikumar and Binoosh, 2020; Ding et al., 2019; Fernández et al., 2020) for online assessment of WMSD risks.

Abobakr et al. (2017) proposed a real-time posture evaluation technique using IMUs and goniometers connected to the upper body of the worker and discussed the effectiveness of feedbacks while performing tasks on postural risks. An IMU measures the linear accelerations (three-axis accelerometer) and rotational velocities (three-axis gyroscope), which can be numerically integrated to obtain the 3-D position/orientation of an object (Razavian et al., 2019). The IMUs were placed on the upper arm, forearm, head, see-through head-mounted display (STHMD), trunk, and pelvis to calculate the segment orientation and limb angles coupled with SG65 goniometers (*Biometrics Ltd*., Newport, UK) to measure wrist angles (flexion/extension, radial/ulnar deviation) (Abobakr et al., 2017). The IMUs are initially calibrated in 2 postures performed to obtain the orientation of the device with respect to the body of the worker. After the initial calibration, the body joint angles are computed from the IMU sensor placed directly on the body part, and RULA assessment is displayed to the worker through STHMD so the worker can understand which body posture is inappropriate. The results of the study showed that real-time ergonomic feedback decreased the risk for musculoskeletal disorders. One of the disadvantages of using IMUs to compute body angles is the occurrence of drift (gradual divergence of calculated position from actual position) due to noise in the integrated signals (Razavian et al., 2019). Furthermore, the IMU sensors and goniometer need to be placed on both sides of the worker's body which can obstruct workers task and makes the experiment setting challenging in many industries (Abobakr et al., 2017).

In order to track upper limb movements for calculating work cycles, Peppoloni et al. (2015) used a wireless, wearable device system that relies on EMG signals to determine muscle effort intensity and on

inertial measurement units (IMUs) to reconstruct the posture of the human upper limb to track upper limb movements for calculating work cycles. The IMU and EMG signals are synchronized using a PLC board that is processed live to a host PC which makes the entire system real-time. The experiment has been conducted on supermarket cashiers processing a bag of ten items weighs between 0.3-7.5kg. The limitation of the study is that the method assumes neck, trunk, and leg score constant in the experiment, which is a prime requirement for calculating Group-B RULA posture score (Peppoloni et al., 2015). In addition, this method requires IMU and EMG sensors to be mounted on a subject's body, but sensor application in real work conditions can be difficult due to signal interference, and trained professionals are required to conduct the study and run calibration procedures, which can also be a challenging process, as threshold parameters of the system vary with a subject's motion during calibration (Peppoloni et al., 2015).

Another method uses marker-less motion capture systems like Microsoft Kinect, which are easy-to-use motion capture devices that can provide real-time anatomical landmark position data in three dimensions (Clark et al., 2012). The experiment was conducted on 20 people at three standing posture settings - single-leg standing balance and the forward and lateral reach and evaluated with the results of a 3D camera-based motion analysis system. The Microsoft Kinect™ method showed excellent concurrent validity with the 3D camera method (Pearson's R-values >0.90) for most measurements (Clark et al., 2012). A significant drawback to this method is the occlusion of body joints in the workplace, which can lead to insufficient information to accurately predict posture and hence, unrealistic results (Clark et al., 2012). Another disadvantage of this method is that it requires wrist, wrist twist, and neck twist RULA scores to be disregarded because the industrial environment produces too much noise and can lead to errors in RULA computation.

A third approach to evaluating postural risk with respect to RULA is to use a computer vision algorithm that predicts the RULA grand score from images. Sasikumar and Binoosh (2020) compared the performance of popular supervised machine learning classifiers such as the Random Forest algorithm, the Naïve Bayes Classifier, the Decision Tree algorithm, the k-Nearest Neighbors algorithm, and Neural Networks and Support Vector Machines when predicting the risk of WMSDs in computer professionals considering postural, physiological, and work-related factors. A preliminary study was conducted to examine the prevalence of WMSDs among 66 I.T professionals using a modified Nordic musculoskeletal questionnaire survey (Crawford, 2007) which was further used to quantify exposure (duration of discomfort or pain), severity (intensity of discomfort or pain during the task) and probability (likelihood of exposure) that leads to WMSDs risk index (Sasikumar and Binoosh, 2020). The WMSDs risk index

was categorized into three levels based on the level of risks to musculoskeletal, which was predicted using the machine learning models. The Random Forest algorithm and Naïve Bayes Classifier predicted the risk of musculoskeletal disorders with the highest accuracy (81.25%) (Sasikumar and Binoosh, 2020). The proposed machine learning algorithms require physiological and work-related factors that are particular to I.T industries as an input attribute to the model for predicting WMSDs risks, which makes the study applicable only for computer intensive tasks. Also, the approach requires the identification of attributes that contributes to WMSDs in an occupational workplace via observation-based tools, which is time-consuming and expensive. Ding et al. (2019) conducted a similar study for real-time assessment of the upper body but for the sitting postures such as in the desk studying or operating a computer. The author assesses the postures by classifying them into pre-defined classes and assign risk scores in accordance with RULA. The classes of the postures are defined based on neck and trunk postures because apparently, they are most vulnerable to WMSDs in computer workers. The study discusses the development of an algorithm that can predict posture classification scores from images using the histogram of oriented gradients (HOG) and Support Vector Machine (SVM) classifier. The HOG is used to extract features from images to capture global and local shape information of the subject, which is then fed into a trained SVM classifier for the classification of posture. The discussed method does not compute wrist score and is limited to sitting postures. In addition to this, the method does not compute individual body part scores or group scores which makes it irrelevant for the ergonomists to identify body parts that are contributing to the risk index. Furthermore, CNN-based models are proven to be more effective than the HOG-SVM for posture estimation that includes occlusion, which is mostly the case in an occupational work posture (Aslan et al., 2020).

Li et al. (2020) conducted a study using a Convolution Neural Network (CNN) to predict joint kinematic information from images and another dense network to classify the output as RULA grand scores in real-time. This method uses the MS COCO dataset as training data for a pose detector model to predict 2D key joint locations for workers. The obtained anatomical keypoints are used as an input for another model to estimate RULA grand score. One major drawback of this technique is that it considers the wrist score uniform throughout the study. The pose detector model is also sensitive to visual noise in images, such as poor lighting conditions and dust, which makes the application difficult in on-site environments. Similarly, Fernández et al. (2020) discuss the use of a pre-trained CNN model to predict RULA scores from the videos but also fails to consider the wrist score. Additionally, the pre-trained model is not able to estimate postures in case of occlusion, which results in failure of RULA score computation due to

lack of information. The CNN model is further trained on synthetic images and validated on the subjects performing the same kind of postures, which causes skeleton detection biases. Another limitation is that the angular measurements are computed from the 2D projection of joints which raises projective distortions in a one-view image. In our study, we are tackling this issue by introducing general guidelines to capture postures from the Camera in 2 views that give enough information about the location of joints and eliminates any projection errors.

In general, all the previous studies related to automating the task of posture assessment either require additional equipment mounted on the body of workers or consider the wrist score uniform for group-A postural evaluation and cannot justifiably be applied broadly to general occupational and industrial tasks (Peppoloni et al., 2015; Plantard et al., 2017; Clark et al., 2012; Li et al., 2020). Thus, the proposed study aims to automate the observation-based process of employing RULA in a workplace by developing a deep learning algorithm that can predict full body posture, including wrist posture, from images without mounting any equipment on workers and evaluate the postural risks associated with occupational tasks. The proposed study has been validated on common occupational postures, and the resulted RULA scores are compared with evaluations by two ergonomic experts. This method reduces the time required for RULA evaluation by eliminating the need for investigators to spend time sampling and evaluating posture from video camera recordings of workplace tasks. Additionally, a license-free web application is developed based on the proposed method and the source-code is made available for public use at https://github.com/OseWindsor/RULA-Posture-Estimation. The web application is intended to aid small and medium size industries, that are not financially equipped to buy high cost objective posture evaluation tools, conduct ergonomic posture assessment. This study aims to determine whether it is possible to develop an automated system based on a deep learning algorithm that will reduce evaluation time and produce RULA score results that are sufficiently similar to those generated by manual evaluators in observational postural assessment.

Chapter 3 – Methodology

3.1 Data Preparation

This study used Whole-Body Human Pose Estimation in the Wild, extending the MS COCO 2017 dataset that was manually annotated for 68 facial points, 42 hand points, and 23 body and feet points for training and validation of the proposed neural network (Jin et al., 2020). The dataset can be found on the official repository available at https://github.com/jin-s13/COCO-WholeBody and available for research and non-commercial purposes as mentioned in the Terms and Conditions. Previously available datasets such as MPII Human Pose Dataset (Andriluka et al., 2015) and Human 3.6M (Ionescu et al., 2014) were manually annotated only for the major body keypoints and, most importantly, lacked wrist joints. Due to this drawback, previously developed machine learning-based full-body posture estimation models implemented multiple cascades of neural network models trained on independent datasets, which increases computational complexity. Also, the data in different datasets vary in terms of illumination, pose, scale, etc., which inevitably introduces dataset biases to the learned parameters, thus hindering the performance of the algorithm from considering the whole-body task (Jin et al., 2020). To the best of our knowledge, the discussed Whole-body COCO dataset is the only publicly available dataset that has manual annotations of the full human body. The annotations have absolute horizontal and vertical distance from the top-left corner of the image for each key point in addition to the visibility flag, which can be 0 (not visible and not available), 1 (available but not visible), and 2 (available and visible). The body keypoints are denoted using fixed indexes in the annotation file for every image, as shown in Fig. 3.1.

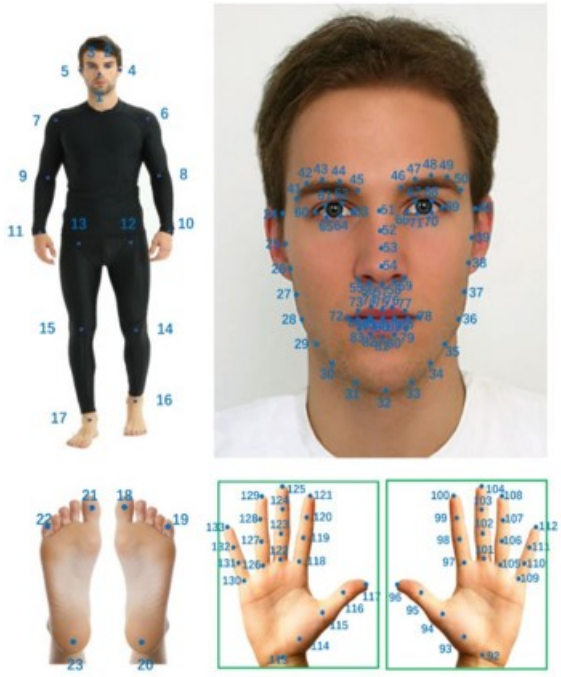

Figure 3.1: Sample Image from Dataset annotated for whole body keypoint index. (Image source: CC by 2.0 license)

Based on RULA notation for whole-body postural analysis, this study used images with visibility flag 1 or 2 for the following 17 key points to train the supervised model: nose (N), right eye (RI), left eye (LI), right shoulder (RS), left shoulder (LS), right elbow (RE), left elbow(LE), right wrist (RW), left wrist (LW), right trunk (RT), left trunk (LT), right knee (RK), left knee (LK), right ankle (RA), left ankle (LA), right knuckle (RN) and left knuckle (LN). The sample image with relevant body joints is shown in Fig. 3.2.

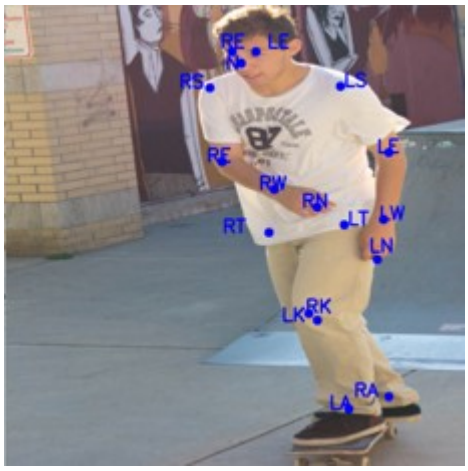

Figure 3.2: Sample Image with annotated keypoints. (*Image source: CC by 2.0 license)*

This study also used a pre-trained neural network, called person detector network, to detect and crop a human figure from an original image in order to reduce the reception area for the postural detection network, thereby improving the accuracy of detection of the postural estimation network. The model weights and model architecture for the pre-trained neural network can be found at https://github.com/experiencor/keras-yolo3, available under an open-source license. The class labels in the script are modified to detect only the 'person' class. The bounding box coordinates extracted from the person detector model were increased from width and height by 5% each to accumulate the full body of the person in case keypoints lie outside the predicted bounding box. Image data augmentation creates new training data out of existing data because machine learning models treats the same image that is rotated differently, and it is almost impossible to capture all the real-world scenario. Data augmentation is a great way to improve the generalization capabilities of the model and avoid overfitting. Data augmentation is applied to generate the vast amount of data required for training CNNs by rotating images at random angles and flipping each along its mid-vertical axis, as shown in Figure 3.3. The key points are transformed accordingly using python script and have been cross-checked randomly by displaying on the screen. This step is necessary to avoid training the supervised learning model with wrong data points. The images in the COCO dataset are of people in different postures engaged in common, real-life activities, which ensured that the model was trained on real data.

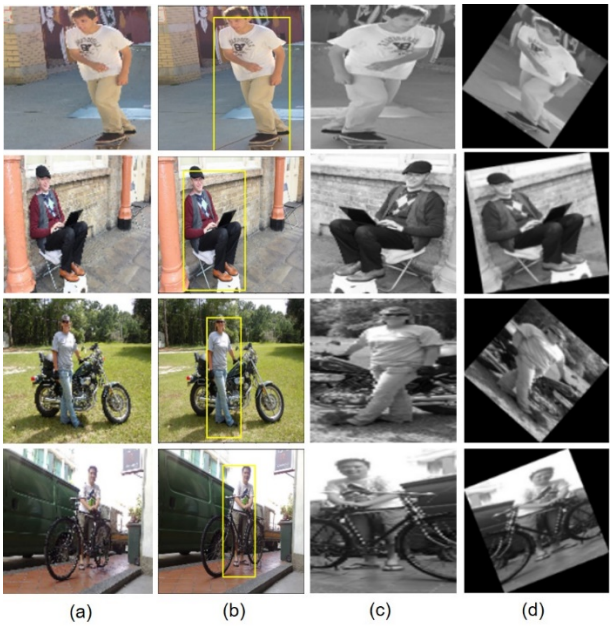

Figure 3.3: (a) Sample training images from MS COCO dataset (b) Results of person detector model (c) Horizontal flip augmentation (d) Random rotation augmentation. (Image source: CC by 2.0 license)

3.2 Network Architecture

A regression-based human pose estimation network was introduced to predict posture from images. The deep learning model in this study was drawn from a study by Toshev et al. (2013) and has been trained using 17 key points from the recently published Whole-Body Human Pose Estimation in the Wild dataset (Jin et al., 2020). The input layer of the network accepts grayscale images of size 128 x 128 pixels. The size of the images selected for training the model was based on the trade-off between memory requirements and model accuracy. An image size of 32x32 pixels would cause the failure of image feature extraction while passing through the model training pipeline due to reduction in dimensionality by MaxPooling layer, whereas an image size of 224x224 pixels would throw memory overflow error in a machine with a GPU size of 25GiB. The selected size of training images increased the speed of model training and reduced the inference time.

Since the joint coordinates are in the absolute image coordinate system and the neural network input layer takes images of size 128 x 128 pixels, images were resized, and the label coordinates transformed relative to the new size of the image. Another reason to resize images was that the images in the dataset vary in size and a CNN model requires an image of fixed size for training. The input features of the model vary between 0-255 (grayscale pixel values), and the output features vary between 0-128 (image coordinates), which led to the processing of features before fed into the model and thus, image standardization was required. The images were normalized by subtracting the mean from each pixel across the channel and then dividing the result by the standard deviation, thereby speeding up convergence during network training.

$$X_S = \frac{X - \mu}{\sigma}$$

Equation 1 *X is the pixel intensity, μ is the mean, and σ is the standard deviation of pixel intensity across the entire channel*

Table 3.1 represents the architecture of the posture estimation network used in this study. The network consists of 5 convolutional layers for feature extraction where first and second consecutive convolutional layers are stacked with a MaxPooling layer for reducing the dimensionality of the feature map and decreasing the number of subsequent trainable parameters (Yamashita et al., 2018). The number of

convolution layers and dense layers are building blocks of the AlexNet architecture which is proven effective in regressing the image pixels to obtain body keypoint location for postures with high occlusion [40]. Internal Covariate Shift is a phenomenon of saturating nonlinearities that slows down the training process by requiring lower learning rates and careful parameter initialization which can be addressed by using the Batch Normalization layer as a part of the model architecture (Ioffe and Szegedy, 2015). The batch Normalization layer transforms the inputs to the convolution layers linearly to have zero mean and unit variance. A Rectified Linear Unit (ReLU) was used as an activation function between each convolutional layer and mapped all negative inputs to zero (Maas et al., 2013). The advantage of using ReLU activation is the non-saturation of the gradients, which accelerates the convergence of gradient optimizers compared to Sigmoid and tanh functions (Krizhevsky et al. 2012). The output feature maps of the final convolutional layer were flattened and connected to 3 fully connected layers with 4096, 4096, and 1000 neurons, respectively, that were connected to the output layer of dimensions 34 x 1. The number of neurons in the output layer was chosen to be 34 because each neuron represents one of the 2D coordinates of 17 key points required for computing RULA scores. The dropout layer forces the neurons to learn features on their own, without depending on other neurons, which makes the CNN model robust. The dropout layer is regularized by the dropout factor, which was chosen after performing the experiment multiple times with different values ranging between 0.1-0.5. A dropout factor greater than 0.5 led to a random drop of more than 50% of the neurons in the network, which increases the generalizability error and increases the overfitting and model complexity (Huang et al., 2017). The dropout layer was implemented with a dropout regularization factor of 0.4 between each fully connected layer which indicated the least difference between validation and training loss. The additional dropout layer reduced the overfitting of the network and improved generalization by randomly dropping out neurons (Srivastava et al., 2014).

Table 3.1: Architecture of postural estimation network

Layer Type	Output Shape	#Parameters
Conv_2D_1(Convolution)	(None, 32, 32, 96)	11712
Bn_1(BatchNormalization)	(None, 32, 32, 96)	384
Max_Pooling_2D_1	(None, 16, 16, 96)	0
Conv_2D_2(Convolution)	(None, 16, 16, 256)	614656
Bn_2(BatchNormalization)	(None, 16, 16, 256)	1024
Max_Pooling_2D_2	(None, 8, 8, 256)	0
Conv_2D_3(Convolution)	(None, 8, 8, 384)	885120
Bn_3(BatchNormalization)	(None, 8, 8, 384)	1536
Conv_2D_4(Convolution)	(None, 8, 8, 384)	1327488
Bn_4(BatchNormalization)	(None, 8, 8, 384)	1536

Conv_2D_5(Convolution)	(None, 8, 8, 256)	884992
Bn_5(BatchNormalization)	(None, 8, 8, 256)	1024
Max_Pooling_2D_3	(None, 4, 4, 256)	0
Flatten	(None, 4096)	0
Dense_1	(None, 4096)	16781312
Bn_6(BatchNormalization)	(None, 4096)	16384
Dropout_1	(None, 4096)	0
Dense_2	(None, 4096)	16781312
Bn_7(BatchNormalization)	(None, 4096)	16384
Dropout_2	(None, 4096)	0
Dense_3	(None, 1000)	4097000
Bn_8(BatchNormalization)	(None, 1000)	4000
Dropout_3	(None, 1000)	0
Dense_4	(None, 34)	34034
Bn_9(BatchNormalization)	(None, 34)	136

The performance of the model during forwarding propagation was calculated using Mean Squared Error as a loss function, and learnable parameters were updated during backpropagation using Adam optimization. Adam is an adaptive optimizer used to tune the learnable parameters of the neural network during backpropagation (Diederik and Ba, 2015). Unlike Gradient Descent optimizers, Adam optimizer does not require manual tuning of learning rate value, although Keras machine learning library allows setting learning rate manually for advanced trials. Adam optimizer adapts the learning rate to the parameters performing small updates for frequently occurring features and large updates for the rarest ones. This allows the neural network to capture information belonging to features that are not frequent and giving them the right weight. Mean Squared Error (MSE) is a model evaluation metric used with regression models to compute the mean of squared prediction errors over all instances in a dataset. MSE is used for computing loss on regression-based models in order to penalize large errors more as compared to small errors, which results in fast convergence of models (Sammut and Webb, 2011). The learning rate is a hyperparameter that controls the amount of change in the model weights due to the computed loss. A high value of learning rate may lead to suboptimal model weights or unstable training process, whereas a learning rate value too small can make the training process very slow. The network was trained with a batch size of 64 and an initial learning rate of 1e-2, which reduces by a factor of 2 up to 1e-8 if the validation loss does not decrease for six consecutive epochs and which stops training when this condition is met. This method of training the neural network is used to prevent the network from overfitting (Caruana et al., 2001). The capability of the network to generalize was assessed by the mean absolute error (MAE) of the model's performance on a validation set during training. MAE is used as an evaluation metric instead of MSE because it gives the absolute error in the prediction made by the neural

network, which is of relevance in this study. MAE is the mean of the absolute difference between the predicted and absolute joint coordinates. An evaluation metric quantifies the performance of the model and doesn't impact the training process. To validate the model and to ensure that training and testing data are representative of the same sample, 20% of the dataset was reserved from the overall training set without any augmentation. The model architecture was developed, trained, and validated using functions provided by the Keras Machine Learning library. The network was trained from scratch using a Google Cloud-based service called Google Colab with 25GiB GPU memory (Bisong, 2019).

3.3 RULA posture score estimation

RULA body posture scores were estimated from 2-D kinematic joint locations obtained from the proposed deep learning model using Euclidean distance and the cosine of the angle between 2D vectors. Euclidean distance was used to calculate the distance between two joints or the length of a limb if the two joints belong to the same body part, and the inverse cosine was used to calculate angles between two limbs.

$$\theta_{ab} = \cos^{-1} \frac{\vec{a}.\vec{b}}{|\vec{a}||\vec{b}|}$$

Equation 2 $\vec{a}$ *and* $\vec{b}$ *are vectors with their respective heads pointing towards a joint and tail lie on the limb connection*.

Angles are denoted by θ with subscripts indicating the specific limbs on either side of θ as denoted by the first letter of the relevant limb, e.g., U for upper arm and L for lower arm followed by a number to distinguish the movement of a single body part. For instance, flexion/extension movement is assigned the number 1, twist movement – 2, side-bending – 3, abduction/adduction movement - 5. S represents scores of a single body part, and its subscripts denote a specific limb and the movement of that limb in accordance with the numerical labels defined above. Figure 3.4 and Figure 3.5 represent RULA posture score calculations for each group from 2D locations of body joints. This study has also incorporated thresholds for some parameters that are necessary to compute RULA scores as given by Vignais et al. (2017).

Upper Arm Flexion/Extension (θ_{U1}): Upper body joints (trunk and shoulder) and upper arm joints (shoulder and elbow) in the sagittal plane were used to calculate upper arm flexion/extension angles.
Upper Arm Abduction (θ_{U5}): Upper Arm was considered elevated if the angle between the upper body

and upper arm was greater than 45º in the frontal plane.

Lower Arm Flexion/Extension (θ_{L1}): Upper arm joints and lower arm joints (wrist and elbow) in the sagittal plane were used to determine lower arm flexion/extension angles.

Lower Arm Midline Posture (S_{L6}): Left and Right Wrist joints are tracked in the frontal plane to estimate whether lower arms were working across the midline.

Wrist Flexion/Extension (θ_{W1}): Lower arm joints (elbow and wrist) and hand joints (wrist and knuckle) in the sagittal plane were used to calculate wrist flexion/extension angles.

Upper Body Leaning (θ_{U4}): Upper Arm score was adjusted by -1 if gravity assisted the posture and required to be initialized by the investigator depending on the nature of the task.

Wrist midline posture (θ_{W6}): Lower arm joints (elbow and wrist) and hand joints (wrist and knuckle) in the frontal plane were used to calculate wrist bending from midline posture. A score of +1 was added to the wrist score when this angle was inferior to -10° (radial deviation) or superior to 10° (ulnar deviation).

Neck Flexion/Extension (θ_{N1}): The neck key point is located at the center of the left and right shoulder key points considering the symmetry of the human body. Upper body joints (trunk and shoulder) and head joints (neck and head) in the sagittal plane were used to calculate neck flexion/extension angle.

Neck Twist (S_{N2}): The Euclidean distances between nose joints and left and right shoulder joints were used in the frontal plane to estimate neck side-twist.

Neck Side-Bending (θ_{N3}): We assumed that the head follows the same orientation as the line connecting the center of the left and right eye to the mid-shoulder joints. Head joints (neck and head) were used to determine neck side-bending with respect to the vertical axis in the frontal plane.

Trunk Flexion/Extension (θ_{T1}): The mid-points of the right and left trunk key points were used to compute the mid-body trunk joint. The neck and mid-body trunk key point positions were used to calculate trunk flexion/extension with respect to the vertical axis in the sagittal plane.

Trunk Twist (S_{T2}): Estimated by the Euclidean distance of shoulder joints from trunk joints in the frontal plane.

Trunk Side-Bending (θ_{T3}): Mid-upper body joint (trunk and neck) positions were used to calculate trunk side bend with respect to the vertical axis in the frontal plane.

Leg Posture (S_{L7}): Legs were assumed to be evenly balanced if the operator was in sitting posture or if the difference between right and left ankle y-coordinates was less than half of torso length in the frontal plane. Although Li and Xu (2019) used a threshold of 5cm, using the torso dimension is preferable because it considers the scale of the person in the image.

Shoulder raises: As the worker did not raise his arm upward during the task, the 'shoulder raising score' was fixed at 0.

Force/Muscle use score: The force and muscle use scores need to be initialized in advance by the investigator according to the nature of the task. For this study, muscle use score and load score were set to +1 because the posture was static and the load intermittent.

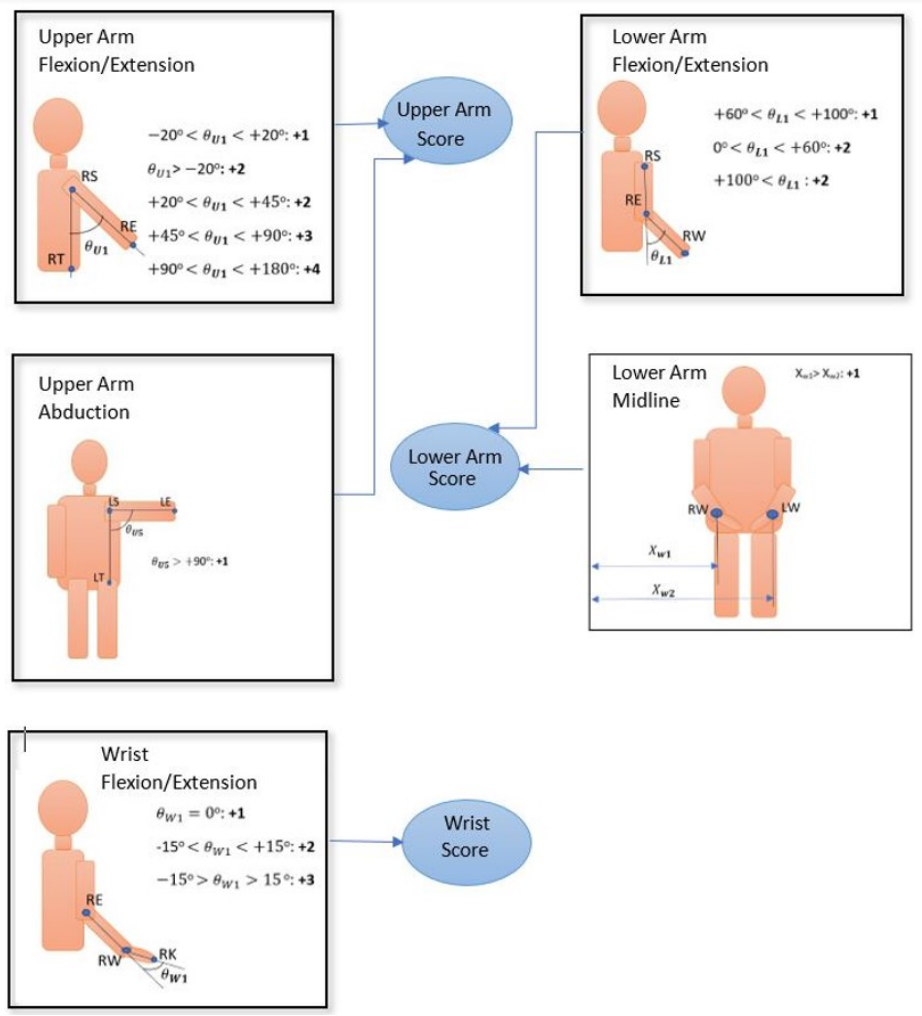

Figure 3.4: Group A postural evaluation technique

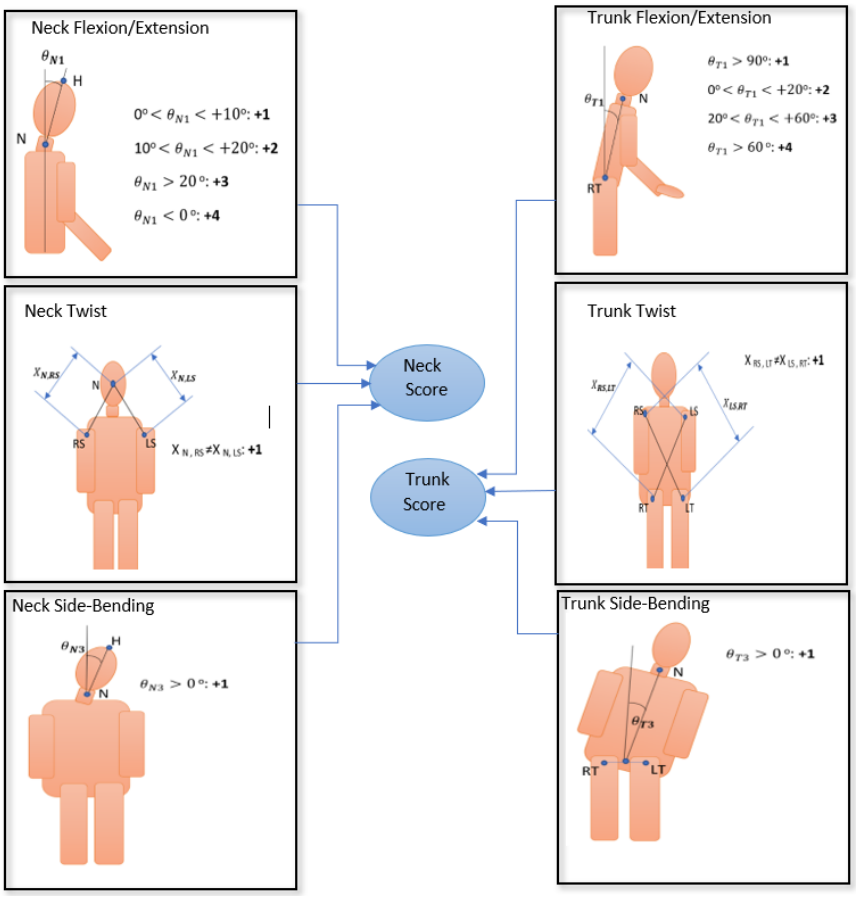

Figure 3.5: Group B postural evaluation technique

3.4 RULA Grand Score Estimation

The Group A score, as determined from the upper arm, lower arm, and wrist scores, and the Group B score, as determined from neck, trunk, and leg scores, combine with the Muscle use score and Force/load score to provide an overall posture score for both groups. Muscle use and load on the body parts were assumed to be constant throughout the work cycle. The scores from both groups were summarized into one action level, called the grand score, which depicts the level of musculoskeletal risk associated with a given body posture and corresponding priority of action associated with that risk, as indicated in Table 3.2.

Table 3.2: RULA grand score summary

Grand Score	Level of WMSDs Risk
1-2	Acceptable working posture if not maintained or repeated for long periods
3-4	Further investigation is needed; posture change may be required

5-6	Investigate and implement posture changes soon to avoid further exposure to WMSDs risk
7	Requires immediate attention and changes in posture

3.5 Validation of the RULA posture score estimation

The posture angles obtained from the algorithm have been validated on 11 common occupational workplace posture images which is shown in Figure 3.7. The postures have been manually evaluated by two ergonomics experts in accordance with RULA. One of the experts computed the angles between body parts for each posture manually by placing a digital goniometer directly on the subject's body during the task for each side, whereas another expert computed the angles from the test images using an online web angle measurement tool (Ginifab – Online Protractor). The images were taken from two different perspectives, one from the side to capture information about joint locations in the sagittal plane and the other from the front to capture information about joint locations in the frontal plane (Fig. 3.6).

Lowe et al. (2014) discuss the guidelines to record postures in a workplace for better quality and accuracy of analysis, and their guidelines have been followed in the current study for test postures in Figure 3.7. The images were captured using a general-purpose camera (Panasonic Lumix DC-ZS70) and contain full-body human postures in indoor and outdoor environments and performing different activities such as pushing, lifting, machining, and handling equipment.

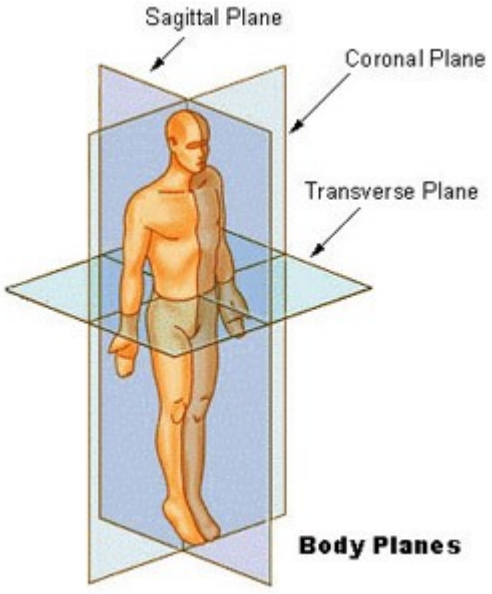

Figure 3.6: Anatomical Planes. (Image source: SEER Training Modules)

The Intraclass Correlation Coefficient (ICC) and Independent sample t-test were applied to the collected

data using SPSS software version 25.0 (IBM Corp., Armonk, NY). The ICC was calculated for the Group A score, Group B score, and Grand Score for both left and right sides to assess the inter-reliability of the postural evaluation between the two rating methods, i.e., manual evaluation and the proposed algorithm. The ICC serves as a reliability index that reflects both degrees of correlation and agreement between the results of two evaluation methods, comparing specifically the ratings from 2 ergonomics experts and those from the proposed machine learning algorithm in this study. The obtained ICC values were computed by a single-rating, absolute agreement, 2-way mixed effect model. This index is used when subjects are chosen at random and raters are fixed, and the difference among the ratings is considered relevant. An independent sample t-test was performed on RULA grand scores obtained by manual technique from the two experts (E1 and E2) and by the proposed algorithm to determine whether any significant difference exists between the new method and the traditional manual approach.

Figure 3.7: Example of test images of postures performing different kinds of occupational tasks from front and side perspective

Chapter 4 – Results

4.1 Model Performance

Figure 4.1 plots the MAE loss for training and validation data computed at every epoch during training of the postural estimation network. The model trained for 187 epochs before stopping early to avoid overfitting. The MAE loss computed at the end of the final epoch on training data is 1.72, whereas the MAE loss on the validation data is 2.86. The trained model weights can be found https://tinyurl.com/model-weights-fbp

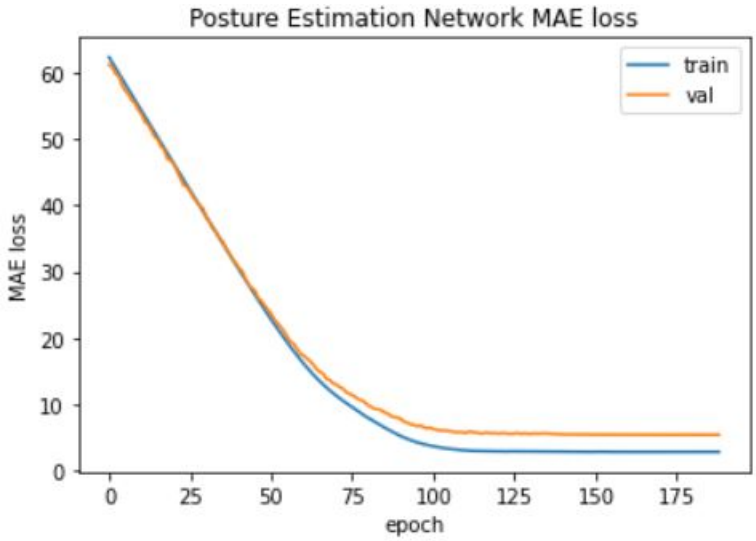

Figure 4.1: MAE plot of posture estimation network during training

4.2 RULA Score Estimation

Table 4.1 indicates the RULA evaluation for Group A, Group B, and Grand Score, along with the inference time of the algorithm for evaluating the postures. The table compares the evaluations from the two experts, denoted E1 and E2, and the results of the proposed automated technique denoted A. Among the computed grand score results for both sides of the body posture, the proposed algorithm assigned the same evaluation scores those of the ergonomics experts to 40.91% of postures, whereas 36.36% of postures were assigned higher evaluation scores, and the remaining 22.73% were assigned lower scores. The average inference time for evaluating the postures using the algorithm is 14.64 seconds.

Table 4.1: Group A, Group B, and Grand Score from RULA postural evaluation for left and right sides of body posture by Expert 1 (E1), Expert 2 (E2), and postural estimation algorithm (A)

Posture	Group A Score						Group B Score						Grand Score						Time (in seconds)	
	Left			Right			Left			Right			Left			Right				
	E1	E2	A	E1	E2	A	E1	E2	A	E1	E2	A	E1	E2	A	E1	E2	A	E	A
1	2	2	2	2	2	2	3	3	2	3	3	2	5	5	4	5	5	4	315	14.01
2	3	3	2	3	3	3	2	2	5	2	2	5	5	5	6	5	5	7	340	13.83
3	3	3	2	4	4	3	5	5	5	5	5	5	7	7	6	7	7	7	410	13.68
4	4	4	3	4	4	4	3	3	3	3	3	3	6	6	6	6	6	6	321	13.77
5	3	3	2	3	3	3	6	6	6	6	6	6	7	7	6	7	7	6	341	14.76
6	2	2	2	2	2	2	2	2	2	2	2	2	4	4	4	4	4	4	337	14.22
7	3	3	3	3	3	3	3	3	3	3	3	3	5	5	6	5	5	6	323	14.26
8	3	3	2	3	3	2	3	3	4	3	3	4	5	5	6	5	5	6	309	14.29
9	4	4	4	4	4	5	2	2	3	2	2	3	6	6	6	6	6	7	327	16.45
10	4	4	4	2	2	2	2	2	2	2	2	2	6	6	6	4	4	4	304	16.01
11	4	4	3	2	2	3	5	5	6	5	5	6	7	7	7	6	6	7	318	15.76

4.3 Statistical Evaluation

The statistical evaluations have been conducted with ratings from one of the manual experts and algorithm instead of the two manual experts. This is because ratings from the two manual evaluators (E1 and E2) are exactly the same for the 11 test postures and considering both in statistical evaluation will unfairly bias the result of statistical tests in favor of the researcher.

An ICC test was conducted with evaluations from one of the experts and algorithms for RULA Group-A scores, RULA group-B scores, and Grand scores for the left and right side of the body posture. As shown in Table 4.2, the ICC index for the Group A score is 0.776 for left-body posture and 0.867 for right-body posture, whereas the Group B score is 0.851 for both sides of the body. Similarly, the ICC index of the computed grand scores are 0.819 and 0.797 for left- and right-body posture, respectively.

Table 4.2: ICC index for Group A Score, Group B Score, and Grand Score with a confidence interval of 95% for left and right-side body postures

Measure	Intraclass Correlation Coefficient (ICC)	
	Left-body Posture	Right-body posture
RULA Group A Score	0.776	0.867
RULA Group B Score	0.851	0.851
RULA Grand Score	0.819	0.797

An independent sample t-test was conducted with evaluations from one of the experts and algorithm as independent variables and grand score as the dependent variable for left and right body sides. As shown in Table 4.3, independent sample t-test results revealed that there is no significant difference in the

evaluation results of left body posture for manual technique (M=5.73, SD=1.009) and proposed technique (M=5.73, SD=0.905); t(20)=0.00, p=1.00. Similarly, there is no significant difference in the evaluation results for the right body posture as well for manual technique (M=5.45, SD=1.036) and proposed technique (M=5.82, SD=1.250); t(20)=-0.743, p=0.466.

Table 4.3: Results from independent sample t-test (Traditional Vs. Proposed Method Evaluation)

Variables	**Manual Evaluation**		**Proposed Method Evaluation**		**t**
	Mean	**SD**	**Mean**	**SD**	
RULA Grand Score - Left	5.73	1.009	5.73	0.905	0.00
RULA Grand Score - Right	5.45	1.036	5.82	1.250	-0.743

Chapter 5 – Discussion

This study introduces a new method to estimate a RULA score from the two views (sagittal and frontal) body posture images with the help of a CNN-based model that is invariant in response to scale, visual noise, and color and can be used to assess postures in workplaces. The subject was recorded using two video cameras throughout the work cycle. One video-camera was positioned to capture information from the side, while another recorded the posture from the front. The performance of the proposed postural estimation network was evaluated by monitoring validation loss in non-augmented split training data. The results of the proposed method were compared with those of the traditional method to assess postural risks based on RULA assessment by computing the ICC as an index for agreement between the two techniques.

The model scored an MAE of 2.86 between predicted and absolute joint coordinates on the validation data, as represented in Figure 4.1, which suggests little error in prediction. Overfitting is a condition in which the network learns the training data too well and subsequently becomes unable to generalize, which results constitute bad performance on new data. The proposed algorithm showed a difference in MAE of 1.14 between the training and validation data, which suggests very little likelihood of neural networks overfitting. Table 4.1 displays the results of RULA evaluation of left- and right-side body posture using manual methods and the proposed algorithm. The proposed algorithm assigned the same evaluation scores as manual evaluators to 40.91% of postures. Although the proportion of identical evaluation scores is relatively low, the algorithm still assigned 77.27% of the postures the same or higher than the corresponding scores achieved with the manual approach. Since postures with higher evaluation scores require more attention from the evaluators, the results indicate that evaluation by the proposed method is more conservative than that of the manual method and thus will have lower omission error than the manual method. This will lead to potentially risky postures being subjected to further investigation in order to minimalize the risk of WMSDs in workplaces.

As shown in Table 4.2, the statistical analysis results demonstrate high levels of agreement between automated and manual evaluation techniques. The ICC values between 0.75 and 0.90 indicate good reliability, and values greater than 0.90 indicate excellent reliability (Koo and Li, 2016). The value of the ICC index is between 0.776 and 0.867 for both sides of body posture in this study, indicating that the algorithm's estimated scores are in good agreement with the investigators' manual evaluation scores. For a few postures, the RULA grand score is different from manual estimation for one side of the body

posture. A possible explanation for this finding is that the upper body posture is asymmetrical between sides in the side-view image and/or that the joint is occluded from other body parts. As mentioned in the literature on RULA, the selection of left- or right-side view can be made based on posture held for the greatest amount of the work cycle, location of the highest loads or a subject's initially reported discomfort (McAtamney and Corlett, 1993). To offset uncertainty caused by asymmetrical posture between left- and right-side views, another camera can be mounted to record both side views.

An independent sample t-test was conducted on RULA grand scores obtained from the experts' manual evaluation and from the proposed algorithm's postural evaluation on both sides of the body. As per the results of the independent sample t-test reported in Table 4.3, there is no significant effect of the evaluation technique on the final RULA Grand score assessment results for both sides of the body posture($p>0.05$). This suggests that the proposed technique can be used as an alternative to the traditional, manual approach in order to evaluate posture based on RULA.

The time taken to evaluate a posture manually is based on factors like posture complexity and investigator experience, whereas the machine learning technique is independent of such factors. As depicted in Table 4.1, the range of inference time for the postural estimation algorithm is 13.68-16.45 seconds as compared to 304-410 seconds by manual technique. The proposed algorithm thus drastically reduces the time taken for postural evaluation and helps ergonomic investigators to make quick decisions. The reported inference time is dependent on the size of the input image and GPU of the machine in which the model is running, which explains the slight difference.

This study makes several contributions to workplace safety and the literature that studies it. The present study is the first to estimate all relevant body posture scores, including the wrist score, from images of postures in the real workplace by implementing machine learning algorithm for observation-based RULA posture assessment. Additionally, the output of the algorithm can be directly interpreted to assess the risks associated with postures, reducing the time and cost typically required to train an ergonomic analyst is conducting a RULA assessment survey. Further, with this method, investigators will no longer need to segment body parts or evaluate posture manually, which will reduce evaluation time and eliminate human error. In addition, the proposed system is easy to use, allowing the investigator to assess postures quickly at the workplace without conducting preliminary surveys to only select postures that were reported for discomfort. Based on the proposed method, this study introduces a web application that allows investigators to upload two-view images of the posture and obtain posture evaluation scores. As shown in Fig 5.1(a), the investigator needs to start the RULA posture evaluation by clicking on "Upload Posture Images" which will load the posture estimation model weights into the system. This

will also prompt the investigator to upload side view image and front view image of the postures; in addition to select few additional parameters necessary to compute RULA scores as shown in Fig 5.1(b). Once the investigator uploads the two-view images and click on the “submit” button, the model will predict the RULA local scores and grand scores for both the sides of the posture and present the result (Fig 5.2). The investigator can interpret level of WMSD risk from the grand scores by referring to the table provided on the same screen. The developed web application will aid the small and medium size industries conduct ergonomic posture assessment without procuring expensive objective posture measurement tools.

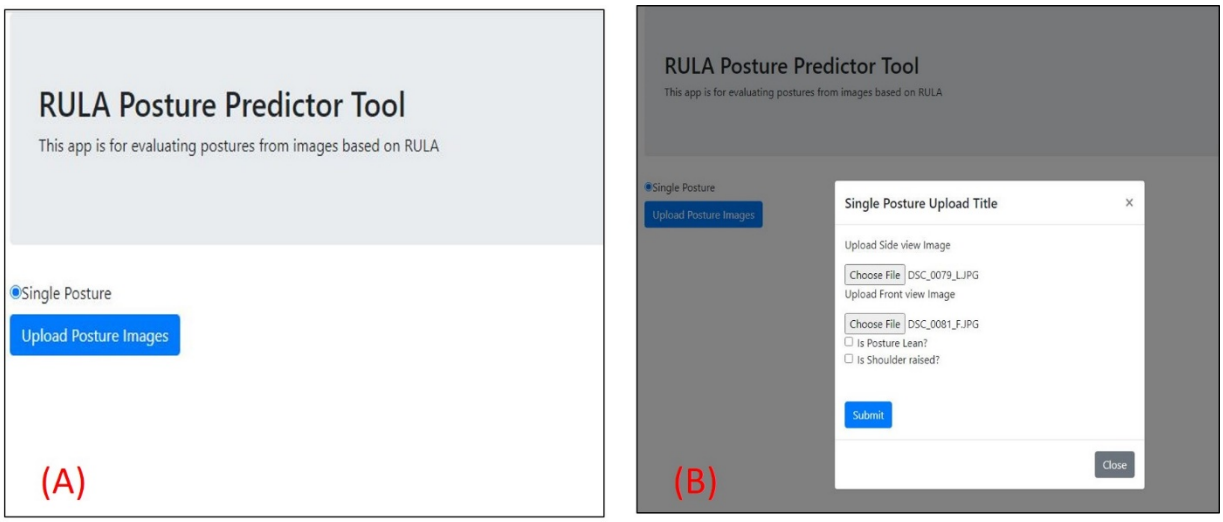

Figure 5.1: RULA Posture Predictor Tool website based on proposed study a) Application Landing Page b) Upload posture screen

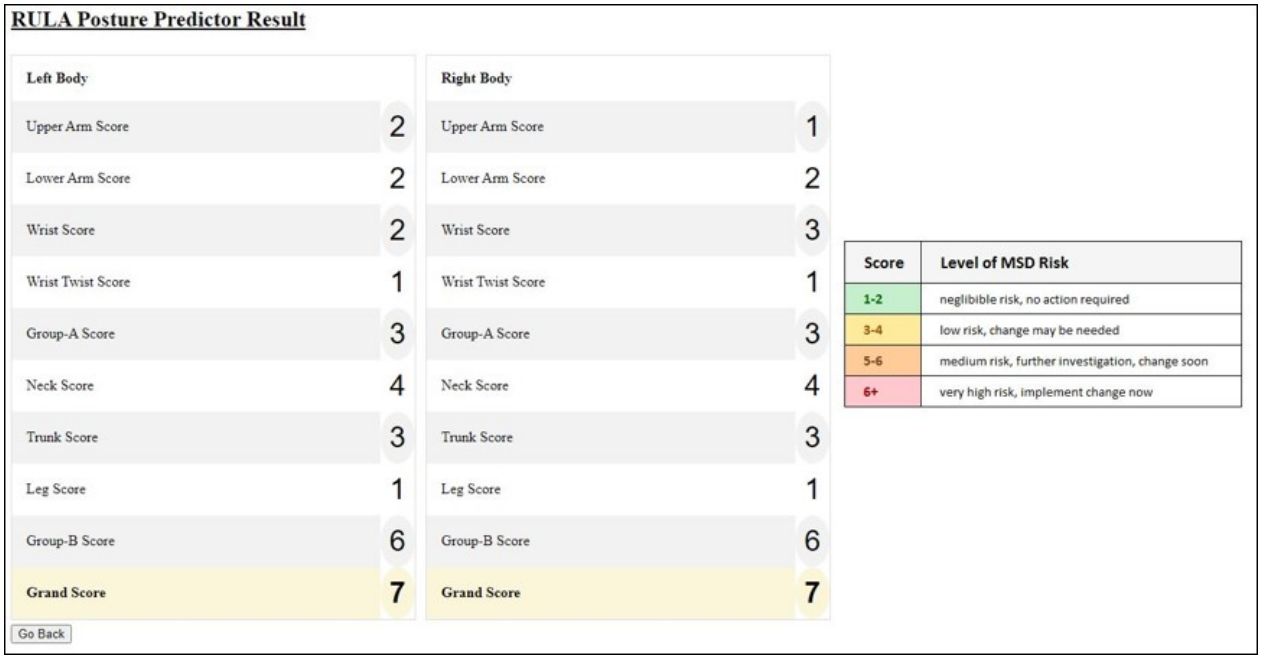

Figure 5.2: RULA based Posture Evaluation result page from the developed system for Posture 11

The last contribution of this study is the consistency and ability to reproduce postural evaluation results due to the negligible contribution of human investigators in the evaluation of posture based on RULA. Based on our results, we conclude that the proposed system will aid ergonomics experts in carrying out the investigation at the workplace and would encourage the use of advanced tools for ergonomic assessments.

There are several features and limitations to this study that should be noted. The first limitation of this study is that the reliability of the algorithm is dependent on the recordings of the postures. Video images represent posture 2-dimensionally and may cause perspective errors if the camera view is not perpendicular to the motion of interest; thus, the postures must be captured such that images depict the true location of body joints (Lowe et al., 2014). The proposed computer vision model can estimate individual group scores from a single view image but requires two images in a perpendicular plane to avoid parallax error in estimating true joint positions and to compute RULA grand scores for postures. Two ergonomic analysts can record postures for the tasks that require movement more than translation in one direction, or a 1 view image can be used to obtain corresponding joint movement such as upper arm flexion/extension from side view image or neck side-bending/twist movement from front view image. The image view required for computing the joint movement is explained in detail in section 3.3. In the future, more advanced neural networks can be incorporated to predict 3D joint locations from the raw image pixels, thereby eliminating the need for images taken from two different orientations

(Martinez et al., 2017; Zhou et al., 2017; Li and Chan, 2015). The second limitation of the study is that the proposed machine learning algorithm is dependent on the person detector model to perform body joint predictions. The person detection model fails on images in which body postures are obstructed heavily with objects. Figure 3.7 shows examples of such images. To overcome this limitation, the posture must be recorded from a better position, which will reduce occlusion and aid the person detector model in detecting the subject. Although it is recommended to capture recordings that most accurately represent the side and front of the posture, the investigator can still alter the position of recording cameras to his own judgment in case the workplace occludes a subject's posture. For instance, Posture in Figure 5.3 is occluded from the vehicle frame to such an extent that the person detector model is unable to detect the subject. However, the same posture can be captured from a different position to obtain results, as shown in Posture 5 in Figure 3.7.

Figure 5.3: Example of posture that the person detection model failed to detect due to high object obstruction

The results of the proposed method were validated against ratings by ergonomic analysts who evaluated the postures using basic ergonomics tools (goniometer and online angle measurement software) to simulate the practical study in the workplace. Although this validation approach could be questioned, this choice was made because recent surveys have proved that ergonomic practitioners worldwide still prefer to use basic tools over direct measurement tools to evaluate postures in workplaces (Lowe et al., 2019). In the future, the results of the proposed method will be compared with the results of the optoelectronic motion capture system, which is gold-standard in objective measurement techniques for posture assessment in the laboratory part of the experiment, which will increase the confidence of readers in the proposed system.

The observation-based posture evaluation method, despite being popular among ergonomics

practitioners, is prone to omission errors that can be effectively overcome with the use of the method deployed in this study. The proposed algorithm has proven to be more conservative in posture evaluations than manual evaluations, which will lead to investigations for ergonomic intervention to discover more postures that pose WMSD risks. The proposed method can evaluate postures without omission error and with performance similar to but faster than manual evaluations. This method, therefore, can aid ergonomic evaluators in conducting RULA-based surveys of occupational postures in workplace conditions.

The posture estimation network discussed in this study can be easily extended to other posture assessment tools like ALLA, REBA and OWAS that evaluate body postures to determine their associated WMSD risks. For instance, REBA evaluates body postures in a similar way to the proposed method by estimating the angles of leg, trunk, neck, arm, and wrist. The output of the postural estimation network is a 2D body joint location that can be processed with a python-based script to evaluate postures based on RULA. This separation of logic allows the proposed postural estimation network to be used for the evaluation of postures based on other tools.

Chapter 6 – Conclusion

This study discusses a DL technique to estimate posture scores from images of postures at occupational workplaces by detecting the 2D kinematic location of body joints, including wrist joints, to evaluate work postures based on RULA. This study proposes a posture estimation network to detect body key point locations of work postures from images. The accuracy of the network has been validated on a subset of training images and achieves high precision. The network has been tested on common workplace postures recorded in indoor and outdoor environments and on workers performing different activities such as lifting, lowering, pushing, etc. The input images are taken from two perspectives, the front, and side of a subject's posture, depicting the true posture in 3D space. The result of the algorithm was compared with manual evaluation from 2 ergonomics experts and validated statistically by computing ICC index and independent sample t-test. The study reported a good degree of agreement in the computed ICC scores between the proposed method and the traditional method, which justifies that the proposed method can be used for posture evaluation as an alternative to manual technique. The independent sample-t test reported that the proposed method can be used as an alternative to the traditional, observation-based approach. The advantage of using DL techniques over other techniques such as using wearable devices or marker-based systems is that it requires no additional setup of instruments in the workplace or any attachable on workers for RULA posture assessment. Although numerous attempts have been made to automate the RULA evaluation using machine learning, the proposed method is the first study that takes wrist posture into consideration while evaluating the work posture, which overcomes the limitation of all the previous literature that consider the wrist score uniform in the study. The automation of observation-based techniques for posture assessment using deep learning will produce results with similar performance as manual evaluators, eliminate errors due to human mistakes, and reduce the time for posture evaluation in workplaces. Additionally, a web application is developed using the method from current study which is made available to public free for use. This will aid small and medium sized industries conduct posture assessment quickly, without being concerned about the high cost incurred with objective tools such as optoelectronic motion measurement system. The proposed method is subjected to the limitation that it requires images of posture from the front and side view to overcoming parallax error of video-based recordings. The real workplace tasks may require motion in more than one direction, and thus, two ergonomic analysts are required to capture posture recordings for such tasks. Additionally, the postures which are heavily occluded from the

workplace may not be detected by the person detector model which require ergonomic analysts to change the position of camera to obtain results. In the future, advanced machine learning algorithms will be developed to estimate postures in 3D space from a single image which will eliminate the need to capture postures from 2 planes. Also, the results of the proposed method need to be evaluated with the results of an objective tool such as an optoelectronic motion measurement system to prove the effectiveness of the study. To further this end, the present study can be extended to evaluate postures based on different postural screening tools such as OWAS, ALLA, REBA etc. The proposed study is expected to encourage ergonomic analysts in conducting RULA-based surveys in the occupational workplaces using advanced tools rather than the traditional observation-based approach. The deep learning-based approach discussed in this study is being able to evaluate posture from images under different viewing conditions i.e., varying illumination and workplace occlusions, which makes the method suitable for real workplace application. WMSDs are the leading cause of work-related disabilities and productivity loss in the workplaces that imposes huge economic burden to developed countries such as Canada and United States. The proposed study will contribute in identifying awkward work postures efficiently which would reduce worker's injuries and save high compensation cost due to WMSDs in the workplaces. The findings of the study is expected to identify workers at risks, thereby helping enhance the health and safety of the workplaces.

References

1. A L Maas, A Y Hannun, A Y Ng. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. Stanford University. http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf
2. Andrews, D. M., Fiedler, K. M., Weir, P. L., & Callaghan, J. P. (2012). The effect of posture category salience on decision times and errors when using observation-based posture assessment methods. Ergonomics, 55(12), 1548–1558. https://doi.org/10.1080/00140139.2012.726656
3. Bernard BP. (1997). Musculoskeletal disorders and workplace factors: a critical review of epidemiologic evidence for work-related musculoskeletal disorders of the neck, upper extremity, and lower back. Centers for Disease control and Prevention, National Institute of Occupational Safety and Health. https://www.cdc.gov/niosh/docs/97-141/
4. Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7
5. Brian D. Lowe, Patrick G. Dempsey, Evan M. Jones. (2019). Ergonomics assessment methods used by ergonomics professionals. Applied Ergonomics. 62(1), 838–842. https://doi.org/10.1016/j.apergo.2019.102882
6. Caruana, R., Lawrence, S., & Giles, L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Neural information processing systems foundation. https://papers.nips.cc/paper/1895-overfitting-in-neural-nets-backpropagation-conjugate-gradient-and-early-stopping.pdf
7. Choobineh, A., Tosian, R., Alhamdi, Z., & Davarzanie, M. (2004). Ergonomic intervention in carpet mending operation. Applied ergonomics, 35(5), 493–496. https://doi.org/10.1016/j.apergo.2004.01.008
8. Clark, R. A., Pua, Y. H., Fortin, K., Ritchie, C., Webster, K. E., Denehy, L., & Bryant, A. L. (2012). Validity of the Microsoft Kinect for assessment of postural control. Gait & posture, 36(3), 372–377. https://doi.org/10.1016/j.gaitpost.2012.03.033
9. Diederik P. Kingma, Jimmy Ba. (2015). Adam: A Method for Stochastic Optimization. International Conference for Learning Representations. https://arxiv.org/abs/1412.6980

10. Elke Schneider, Xabier Irastorza and Sarah Copsey. (2010). European Agency for Safety and Health at Work. Occupational Safety and Health Administration. https://osha.europa.eu/en/publications/reports/TERO09009ENC
11. G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger. (2017). Densely Connected Convolutional Networks. IEEE Conference on Computer Vision and Pattern Recognition. pp. 2261-2269. Retrieved from: https://arxiv.org/abs/1608.06993
12. González, B., Adenso-Díaz, B., & Torre, P. (2003). Ergonomic performance and quality relationship: an empirical evidence case. International Journal of Industrial Ergonomics, 31, 33-40. https://doi.org/10.1016/S0169-8141(02)00116-6
13. Hignett, S., & McAtamney, L. (2000). Rapid entire body assessment (REBA). Applied ergonomics, 31(2), 201–205. https://doi.org/10.1016/s0003-6870(99)00039-3
14. Ionescu, C., Papava, D., Olaru, V., & Sminchisescu, C. (2014). Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. IEEE transactions on pattern analysis and machine intelligence, 36(7), 1325–1339. https://doi.org/10.1109/TPAMI.2013.248
15. Jas and Fred Fang, 2020. GitHub. COCO-WholeBody. https://github.com/jin-s13/COCO-WholeBody
16. Jin S. et al. (2020) Whole-Body Human Pose Estimation in the Wild. Computer Vision – ECCV 2020, vol 12354. https://doi.org/10.1007/978-3-030-58545-7_12
17. Joanne O. Crawford. (2007). The Nordic Musculoskeletal Questionnaire, *Occupational Medicine*. 57(4), 300–301. https://doi.org/10.1093/occmed/kqm036
18. Julieta Martinez, Rayat Hossain, Javier Romero, James J. Little. 2017. A simple yet effective baseline for 3d human pose estimation. Computer Vision and Pattern Recognition. https://arxiv.org/abs/1705.03098
19. Karhu, O., Kansi, P., & Kuorinka, I. (1977). Correcting working postures in industry: A practical method for analysis. Applied ergonomics, 8(4), 199–201. https://doi.org/10.1016/0003-6870(77)90164-8
20. Kee D. (2020). An empirical comparison of OWAS, RULA and REBA based on self-reported discomfort. International journal of occupational safety and ergonomics: JOSE, 26(2), 285–295. https://doi.org/10.1080/10803548.2019.1710933

21. Kong, Y. K., Lee, S. Y., Lee, K. S., & Kim, D. M. (2018). Comparisons of ergonomic evaluation tools (ALLA, RULA, REBA and OWAS) for farm work. International journal of occupational safety and ergonomics: JOSE, 24(2), 218–223. https://doi.org/10.1080/10803548.2017.1306960
22. Koo, T. K., & Li, M. Y. (2016). A Guideline of Selecting and Reporting Intraclass Correlation Coefficients for Reliability Research. Journal of chiropractic medicine, 15(2), 155–163. https://doi.org/10.1016/j.jcm.2016.02.012
23. L. Peppoloni, A. Filippeschi, E. Ruffaldi, C.A. Avizzano. (2015). A novel wearable system for the online assessment of risk for biomechanical load in repetitive efforts. International Journal of Industrial Ergonomics. pp. 1-11. https://doi.org/10.1016/j.ergon.2015.07.002
24. Li Li, Tara Martin, Xu Xu, 2020. A novel vision-based real-time method for evaluating postural risk factors associated with musculoskeletal disorders. Applied Ergonomics 87:103138. https://doi.org/10.1016/j.apergo.2020.103138
25. Li S., Chan A.B. (2015) 3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. Computer Vision -- ACCV 2014. https://doi.org/10.1007/978-3-319-16808-1_23
26. Lin TY. et al. (2014) Microsoft COCO: Common Objects in Context. Computer Vision – ECCV 2014, vol 8693. https://doi.org/10.1007/978-3-319-10602-1_48
27. Massaccesi, M., Pagnotta, A., Soccetti, A., Masali, M., Masiero, C., & Greco, F. (2003). Investigation of work-related disorders in truck drivers using RULA method. Applied ergonomics, 34(4), 303–307. https://doi.org/10.1016/S0003-6870(03)00052-8
28. Matt Middlesworth. (2020). The Cost of Musculoskeletal Disorders (MSDs). ErgoPlus. https://ergo-plus.com/cost-of-musculoskeletal-disorders-infographic/
29. McAtamney, L., & Nigel Corlett, E. (1993). RULA: a survey method for the investigation of work-related upper limb disorders. Applied ergonomics, 24(2), 91–99. https://doi.org/10.1016/0003-6870(93)90080-s
30. Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler and Bernt Schiele (2015). Max Planck Institute for Informatics: MPII Human Pose Dataset. http://human-pose.mpi-inf.mpg.de/
31. Plantard, P., Shum, H., Le Pierres, A. S., & Multon, F. (2017). Validation of an ergonomic assessment method using Kinect data in real workplace conditions. Applied ergonomics, 65, 562–569. https://doi.org/10.1016/j.apergo.2016.10.015

32. Ryan G. A. (1989). The prevalence of musculo-skeletal symptoms in supermarket workers. Ergonomics, 32(4), 359–371. https://doi.org/10.1080/00140138908966103
33. Sammut C., Webb G.I. (2011). Mean Squared Error. Encyclopedia of Machine Learning. https://doi.org/10.1007/978-0-387-30164-8_528
34. Sanchez-Lite, A et al. (2013). Novel ergonomic postural assessment method (NERPA) using product-process computer aided engineering for ergonomic workplace design. PloS one, 8(8), e72703. https://doi.org/10.1371/journal.pone.0072703
35. Sasikumar, V., & Binoosh, S. (2020). A model for predicting the risk of musculoskeletal disorders among computer professionals. International journal of occupational safety and ergonomics: JOSE, 26(2), 384–396. https://doi.org/10.1080/10803548.2018.1480583
36. Sebbag E, Felten R, Sagez F, et al. (2019). The world-wide burden of musculoskeletal diseases: a systematic analysis of the World Health Organization Burden of Diseases Database. Annals of the Rheumatic Diseases. 78: 844-848. Doi: 10.1136/annrheumdis-2019-215142
37. Serge Simoneau, Marie St-Vincent and Denise Chicoine. (1996). Institut de recherche Robert-Sauvé en santé et en sécurité du travail (IRSST). Work Related Musculoskeletal Disorders (WMSDs). Retrieved from: https://www.irsst.qc.ca/media/documents/PubIRSST/RG-126-ang.pdf
38. Sergey Ioffe and Christian Szegedy. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning. 37 (ICML'15). JMLR.org, 448–456. https://arxiv.org/abs/1502.03167
39. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958. https://dl.acm.org/doi/abs/10.5555/2627435.2670313
40. Toshev, Alexander & Szegedy, Christian. (2013). DeepPose: Human Pose Estimation via Deep Neural Networks. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2014.214
41. Tuure V-M. (1992). Determination of physical stress in agricultural work. International Journal of Industrial Ergonomics. https://doi.org/10.1016/0169-8141(92)90094-G
42. Van Wely P. (1970). Design and disease. Applied ergonomics, 1(5), 262–269. https://doi.org/10.1016/0003-6870(70)90075-x

43. Van Wendel de Joode, B., Burdorf, A., & Verspuy, C. (1997). Physical load in ship maintenance: hazard evaluation by means of a workplace survey. Applied ergonomics, 28(3), 213–219. https://doi.org/10.1016/s0003-6870(96)00051-8
44. Vern Putz-Anderson, et.al. (1997). Musculoskeletal Disorders and Workplace Factors. Centers for Disease Control and Prevention. https://www.cdc.gov/niosh/docs/97-141/pdfs/97-141.pdf
45. Wright EJ, Haslam RA. (1999). Manual handling risks and controls in a soft drink distribution centre. Applied Ergonomics. 30:311–8. https://doi.org/10.1016/S0003-6870(98)00036-2
46. Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, Yichen Wei. (2017). Towards 3D Human Pose Estimation in the Wild: Weakly-supervised Approach. Computer Vision and Pattern Recognition. https://arxiv.org/abs/1704.02447
47. Yamashita, R., Nishio, M., Do, R.K.G. et al. (2018). Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629. https://doi.org/10.1007/s13244-018-0639-9
48. Butterworth-Heinemann. (1974). Applied Ergonomics Handbook. Science Direct: ISBN 9780902852389. https://www.sciencedirect.com/book/9780902852389/applied-ergonomics-handbook
49. Yale Environmental Health and Safety. (2018). Ergonomics: Awkward Posture. Retrieved from: https://ehs.yale.edu/sites/default/files/files/ergo-awkward-posture.pdf.
50. Work-related Musculoskeletal Disorders (WMSDs). (2014). Canadian Centre for Occupational Health and Safety. Retrieved from: https://www.ccohs.ca/oshanswers/diseases/rmirsi.html.
51. David G. C. (2005). Ergonomic methods for assessing exposure to risk factors for work-related musculoskeletal disorders. Occupational medicine (Oxford, England), 55(3), 190–199. https://doi.org/10.1093/occmed/kqi082
52. Pope D, Silman A, Cherry N, Pritchard C, MacFarlane G. (1998). Validity of a self-completed questionnaire measuring the physical demands of work. Scand J Work Environ Health; 24:376 – 385.
53. Van der Beek A, Frings-Dressen M. (1998). Assessment of mechanical exposure in ergonomic epidemiology. Occup Environ Med; 55:291 –299. http://dx.doi.org/10.1136/oem.55.5.291

54. Tzu-Hsien Lee & Chia-Shan Han (2013) Analysis of Working Postures at a Construction Site Using the OWAS Method, International Journal of Occupational Safety and Ergonomics, 19:2, 245-250, DOI: 10.1080/10803548.2013.11076983

55. Gómez-Galán, M., Pérez-Alonso, J., Callejón-Ferre, Á. J., & López-Martínez, J. (2017). Musculoskeletal disorders: OWAS review. Industrial health, 55(4), 314–337. https://doi.org/10.2486/indhealth.2016-0191

56. Takala, E. P., Pehkonen, I., Forsman, M., Hansson, G. A., Mathiassen, S. E., Neumann, W. P., Sjøgaard, G., Veiersted, K. B., Westgaard, R. H., & Winkel, J. (2010). Systematic evaluation of observational methods assessing biomechanical exposures at work. Scandinavian journal of work, environment & health, 36(1), 3–24. https://doi.org/10.5271/sjweh.2876

57. Justavino, F. & Ramirez, R. & Perez, N. & Borz, Stelian. (2015). THE use of OWAS in forest operations postural assessment: Advantages and limitations. 8. 7-16. https://www.semanticscholar.org/paper/The-use-of-OWAS-in-forest-operations-postural-and-Justavino-Ramirez/39da993fe36d6ec4ac70341a30201d27df7f0b17

58. Engels, J. A., Landeweerd, J. A., & Kant, Y. (1994). An OWAS-based analysis of nurses' working postures. Ergonomics, 37(5), 909–919. https://doi.org/10.1080/00140139408963700

59. Hignett S. (1994) Using computerized OWAS for postural analysis of nursing work. Ergonomics-Societys, 1994 Annual Conference-Contemporary Ergonomics 1994. 253–258. https://doi.org/10.2486/indhealth.2016-0191

60. Das, B., & Gangopadhyay, S. (2015). Prevalence of musculoskeletal disorders and physiological stress among adult, male potato cultivators of West Bengal, India. Asia-Pacific journal of public health, 27(2), NP1669–NP1682. https://doi.org/10.1177/1010539511421808

61. Banibrata Das, Tirthankar Ghosh & Somnath Gangopadhyay (2012) Assessment of Ergonomic and Occupational Health-Related Problems Among Female Prawn Seed Collectors of Sunderbans, West Bengal, India, International Journal of Occupational Safety and Ergonomics, 18:4, 531-540, DOI: 10.1080/10803548.2012.11076949

62. Das, B., Ghosh, T., & Gangopadhyay, S. (2013). Child work in agriculture in West Bengal, India: assessment of musculoskeletal disorders and occupational health problems. Journal of occupational health, 55(4), 244–258. https://doi.org/10.1539/joh.12-0185-oa

63. Abaraogu, U. O., Odebiyi, D. O., & Olawale, O. A. (2016). Association between postures and work-related musculoskeletal discomforts (WRMD) among beverage bottling workers. Work (Reading, Mass.), 54(1), 113–119. https://doi.org/10.3233/WOR-162262

64. Yanes Escalona, L., Sandia Venot, R., Escalona, E., & Yanes, L. (2012). The reality of the women who make our lives easier: experience in a company that assemblies electric motors in Venezuela. Work (Reading, Mass.), 41 Suppl 1, 1775–1777. https://doi.org/10.3233/WOR-2012-0384-1775

65. Gallo, R., & Mazzetto, F. (2013). Ergonomic analysis for the assessment of the risk of work-related musculoskeletal disorder in forestry operations. Journal of Agricultural Engineering, 44(s2). https://doi.org/10.4081/jae.2013.389

66. Houshyar, E., & Kim, I. (2018). Understanding musculoskeletal disorders among Iranian apple harvesting laborers: Ergonomic and stopwatch time studies. International Journal of Industrial Ergonomics, 67, 32-40. https://doi.org/10.1016/j.ergon.2018.04.007

67. Enez, K., & Nalbantoğlu, S.S. (2019). Comparison of ergonomic risk assessment outputs from OWAS and REBA in forestry timber harvesting. International Journal of Industrial Ergonomics, 70, 51-57. https://doi.org/10.1016/j.ergon.2019.01.009

68. Asadi, H., Yu, D., & Mott, J. (2019). Risk factors for musculoskeletal injuries in airline maintenance, repair & overhaul. International Journal of Industrial Ergonomics, 70, 107-115. https://doi.org/10.1016/j.ergon.2019.01.008

69. Diego-Mas, J.-A., Alcaide-Marzal, J., & Poveda-Bautista, R. (2017). Errors Using Observational Methods for Ergonomics Assessment in Real Practice. Human Factors, 59(8), 1173–1187. https://doi.org/10.1177/0018720817723496

70. Dumas, G. A., Preston, D., Beaucage-Gauvreau, E., & Lawani, M. (2014). Posture analysis of lifting a load for head carriage and comparison between pregnant and non-pregnant women. Work (Reading, Mass.), 47(1), 63–72. https://doi.org/10.3233/WOR-131686

71. Das, B. (2015). An evaluation of low back pain among female brick field workers of West Bengal, India. Environ Health Prev Med 20, 360–368. https://doi.org/10.1007/s12199-015-0476-0

72. Work-related Musculoskeletal Disorders (WMSDs) - Risk Factors, 2014. Canadian Centre for Occupational Health and Safety. https://www.ccohs.ca/oshanswers/ergonomics/risk.html.

73. Yazdanirad, S., Khoshakhlagh, A. H., Habibi, E., Zare, A., Zeinodini, M., & Dehghani, F. (2018). Comparing the Effectiveness of Three Ergonomic Risk Assessment Methods-RULA, LUBA, and NERPA-to Predict the Upper Extremity Musculoskeletal Disorders. Indian journal of occupational and environmental medicine, 22(1), 17–21. https://doi.org/10.4103/ijoem.IJOEM_23_18

74. Huang, T. (1996). Computer Vision: Evolution And Promise. https://www.semanticscholar.org/paper/Computer-Vision%3A-Evolution-And-Promise-Huang/0390930e75300a877f0064415a220da421edbb0c

75. J. Aloimonos, 1990. Purposive and qualitative active vision. Proceedings 10th International Conference on Pattern Recognition. doi: 10.1109/ICPR.1990.118128

76. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94

77. Calonder M., Lepetit V., Strecha C., Fua P. (2010) BRIEF: Binary Robust Independent Elementary Features. Computer Vision – ECCV 2010. vol 6314. https://doi.org/10.1007/978-3-642-15561-1_56

78. Mahony, N.O., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco-Hernández, G., Krpalkova, L., Riordan, D., & Walsh, J. (2019). Deep Learning vs. Traditional Computer Vision. CVC. DOI: 10.1007/978-3-030-17795-9

79. O' Mahony N, Murphy T, Panduru K, et al (2016) Adaptive process control and sensor fusion for process analytical technology. In: 2016 27th Irish Signals and Systems Conference (ISSC). IEEE, pp 1–6. DOI: 10.1109/ISSC.2016.7528449

80. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5, 115–133. doi:10.1007/ BF02478259

81. Gallo, Crescenzio. (2015). Artificial Neural Networks: tutorial. https://www.researchgate.net/publication/261392616_Artificial_Neural_Networks_tutorial

82. MIT Press. (2014). Introduction to machine learning. Cambridge. https://mitpress.mit.edu/books/introduction-machine-learning

83. Liu Q., Wu Y. (2012) Supervised Learning. In: Seel N.M. (eds) Encyclopedia of the Sciences of Learning. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-1428-6_451

84. Siadati, Saman. (2018). What is UnSupervised Learning. 10.13140/RG.2.2.33325.10720.

85. Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. Machine Learning Proceedings

86. Shweta Bhatt. (2018). 5 Things You Need to Know about Reinforcement Learning. KDNuggets. https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html.

87. Malmgren-Hansen, D., Engholm, R., & Pedersen, M.Ø. (2016). Training Convolutional Neural Networks for Translational Invariance on SAR ATR. https://ieeexplore.ieee.org/document/7559339/citations#citations

88. Yang, S., Luo, P., Loy, C. C., & Tang, X. (2018). Faceness-Net: Face Detection through Deep Facial Part Responses. IEEE transactions on pattern analysis and machine intelligence, 40(8), 1845–1859. https://doi.org/10.1109/TPAMI.2017.2738644

89. Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep Learning Face Attributes in the Wild. 2015 IEEE International Conference on Computer Vision (ICCV), 3730-3738. https://ieeexplore.ieee.org/document/7410782

90. Köstinger, M., Wohlhart, P., Roth, P., & Bischof, H. (2011). Annotated Facial Landmarks in the Wild: A large-scale, real-world database for facial landmark localization. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2144-2151. https://ieeexplore.ieee.org/abstract/document/6130513

91. J. Redmon, S. Divvala, R. Girshick and A. Farhadi, 2016. You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 779-788, doi: 10.1109/CVPR.2016.91.

92. Russakovsky, O., Deng, J., Su, H. et al. (2015). ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis 115, 211–252. https://doi.org/10.1007/s11263-015-0816-y

93. Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE transactions on pattern analysis and machine intelligence, 39(6): 1137–49. https://arxiv.org/abs/1506.01497

94. Kamate, Shreyamsh & Yilmazer, Nuri. (2015). Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles. Procedia Computer Science. 61. 436-441. 10.1016/j.procs.2015.09.183.

95. Chen, J., Ahn, C., & Han, S. (2014). Detecting the Hazards of Lifting and Carrying in Construction through a Coupled 3D Sensing and IMUs Sensing System. DOI: 10.1061/9780784413616.138
96. Azhar, Muhamad. (2014). Moving Object Detection in Industrial Line Application. 10.13140/RG.2.2.32398.05448.
97. Gene Lewis, 2016. Object Detection for Autonomous Vehicles. Stanford University. https://web.stanford.edu/class/cs231a/prev_projects_2016/object-detection-autonomous.pdf
98. Hu, Y., Li, Y., Yang, T., & Pan, Q. (2018). Short Text Classification with A Convolutional Neural Networks Based Method. 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), 1432-1435. https://ieeexplore.ieee.org/document/8581332
99. Unstructured Text. (n.d.). Science Direct. https://www.sciencedirect.com/topics/mathematics/unstructured-text.
100. Movie Review Data. (n.d.). Cornell University. Data (cornell.edu)
101. CAP5415-Computer Vision Lecture 21-Deformable Part Model (DPM). Computer Vision. Retrieved from: https://www.cs.ucf.edu/~bagci/teaching/computervision16/Lec21.pdf.
102. Yang, Yi & Ramanan, Deva. (2013). Articulated Human Detection with Flexible Mixtures of Parts. IEEE transactions on pattern analysis and machine intelligence. 35. 2878-90. 10.1109/TPAMI.2012.261.
103. Carreira, J., Agrawal, P., Fragkiadaki, K., & Malik, J. (2016). Human Pose Estimation with Iterative Error Feedback. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4733-4742. https://arxiv.org/abs/1507.06550
104. Johnson, S., & Everingham, M. (2010). Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. BMVC. doi:10.5244/C.24.12
105. Newell, A., Yang, K., & Deng, J. (2016). Stacked Hourglass Networks for Human Pose Estimation. ECCV. https://arxiv.org/abs/1603.06937
106. Sapp, B., Taskar, B. (2013). Modec: Multimodal decomposable models for human pose estimation. In: Computer Vision and Pattern Recognition. CVPR. https://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Sapp_MODEC_Multimodal_Decomposable_2013_CVPR_paper.pdf
107. Park, S., Hwang, J., & Kwak, N. (2016). 3D Human Pose Estimation Using Convolutional Neural Networks with 2D Pose Information. ECCV Workshops. https://arxiv.org/abs/1608.03075

108. Abobakr, A., Nahavandi, D., Iskander, J., Hossny, M., Nahavandi, S., & Smets, M. (2017). RGB-D human posture analysis for ergonomie studies using deep convolutional neural network. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2885-2890. https://doi.org/10.1109/SMC.2017.8123065
109. Vignais, N., Miezal, M., Bleser, G., Mura, K., Gorecky, D., & Marin, F. (2013). Innovative system for real-time ergonomic feedback in industrial manufacturing. Applied ergonomics, 44 4, 566-74. https://doi.org/10.1016/j.apergo.2012.11.008
110. Li, P., Meziane, R., Otis, M., Ezzaidi, H., & Cardou, P. (2014). A Smart Safety Helmet using IMU and EEG sensors for worker fatigue detection. 2014 IEEE International Symposium on Robotic and Sensors Environments (ROSE) Proceedings, 55-60. https://ieeexplore.ieee.org/document/6952983
111. Yan, X., Li, H., Li, A.R., & Zhang, H. (2017). Wearable IMU-based real-time motion warning system for construction workers' musculoskeletal disorders prevention. Automation in Construction, 74, 2-11. https://doi.org/10.1016/j.autcon.2016.11.007
112. Reza Sharif Razavian, Sara Greenberg, and John McPhee. (2019). Biomechanics Imaging and Analysis. Encyclopedia of Biomedical Engineering. 488--500. https://www.elsevier.com/books/encyclopedia-of-biomedical-engineering/narayan/978-0-12-804829-0
113. Morbidity and Mortality Weekly Report (MMWR) (2013). Centers of Disease Control and Prevention. Retrieved from: https://www.cdc.gov/mmwr/preview/mmwrhtml/mm6222a2.htm.
114. Macpherson, R. A., Lane, T. J., Collie, A., & McLeod, C. B. (2018). Age, sex, and the changing disability burden of compensated work-related musculoskeletal disorders in Canada and Australia. BMC public health, 18(1), 758. https://doi.org/10.1186/s12889-018-5590-7
115. Work-Related Musculoskeletal Disorders & Ergonomics. (n.d.). Centers of Disease Control and Prevention. https://www.cdc.gov/workplacehealthpromotion/health-strategies/musculoskeletal-disorders/index.html.
116. Fazi, H.B., Mohamed, N.M., & Basri, A.Q. (2019). Risks assessment at automotive manufacturing company and ergonomic working condition. https://iopscience.iop.org/article/10.1088/1757-899X/469/1/012106
117. Vignais, N., Bernard, F., Touvenot, G., & Sagot, J. C. (2017). Physical risk factors identification based on body sensor network combined to videotaping. Applied ergonomics, 65, 410–417. https://doi.org/10.1016/j.apergo.2017.05.003

118. Li, L., & Xu, X. (2019). A deep learning-based RULA method for working posture assessment. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 63, 1090 - 1094. https://doi.org/10.1177/1071181319631174
119. Lowe, B., Weir, P., & Andrews, D. (2014). Observation-based posture assessment: review of current practice and recommendations for improvement. https://www.cdc.gov/niosh/docs/2014-131/default.html
120. Ding, Z., Li, W., Ogunbona, P. et al. (2019). A real-time webcam-based method for assessing upper-body postures. Machine Vision and Applications 30, 833–850. https://doi.org/10.1007/s00138-019-01033-9
121. Yang, K., Ahn, C., & Kim, H. (2020). Deep learning-based classification of work-related physical load levels in construction. Adv. Eng. Informatics, 45, 101104. https://doi.org/10.1016/j.aei.2020.101104
122. MassirisFernández, M., Fernández, J.Á., Bajo, J.M., & Delrieux, C. (2020). Ergonomic risk assessment based on computer vision and machine learning. Comput. Ind. Eng., 149, 106816. https://doi.org/10.1016/j.cie.2020.106816
123. Aslan, M.F., Durdu, A., Sabanci, K., & Mutluer, M.A. (2020). CNN and HOG based comparison study for complete occlusion handling in human tracking. Measurement, 158, 107704. https://doi.org/10.1016/j.measurement.2020.107704
124. Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

Appendices

Necessary Imports of Libraries

```
import os
import scipy.io
import scipy.misc

import PIL
import struct

from skimage.transform import resize

from keras.layers.merge import add, concatenate

import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
from matplotlib.patches import Rectangle

%matplotlib inline

from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array

import keras
from keras import Input, metrics
from keras.layers import MaxPooling2D, Conv2D, GlobalAveragePooling2D, Convolu
tion2D, Flatten, Dense, Dropout, LeakyReLU, BatchNormalization, Activation
from keras.layers import Add, ZeroPadding2D, AveragePooling2D, MaxPooling2D, G
lobalMaxPooling2D, Lambda, LeakyReLU, UpSampling2D
from keras.models import Model, Sequential, load_model, model_from_json
from keras.preprocessing import image
from keras.callbacks import ModelCheckpoint

import numpy as np
import random
import math
from numpy import clip, expand_dims, asarray
import cv2

import pandas as pd
import time
from google.colab.patches import cv2_imshow
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
from PIL import Image
```

```
import pickle as pkl
from os import listdir
from os.path import isfile, join
```

Functions for building Pre-Trained person detector Neural network

```
class WeightReader:
  def __init__(self, weight_file):
    with open(weight_file, 'rb') as w_f:
      major,  = struct.unpack('i', w_f.read(4))
      minor,  = struct.unpack('i', w_f.read(4))
      revision, = struct.unpack('i', w_f.read(4))
      if (major*10 + minor) >= 2 and major < 1000 and minor < 1000:
        w_f.read(8)
      else:
        w_f.read(4)
      transpose = (major > 1000) or (minor > 1000)
      binary = w_f.read()
    self.offset = 0
    self.all_weights = np.frombuffer(binary, dtype='float32')

  def read_bytes(self, size):
    self.offset = self.offset + size
    return self.all_weights[self.offset-size:self.offset]

  def load_weights(self, model):
    for i in range(106):
      try:
        conv_layer = model.get_layer('conv_' + str(i))
        print("loading weights of convolution #" + str(i))
        if i not in [81, 93, 105]:
          norm_layer = model.get_layer('bnorm_' + str(i))
          size = np.prod(norm_layer.get_weights()[0].shape)
          beta  = self.read_bytes(size) # bias
          gamma = self.read_bytes(size) # scale
          mean  = self.read_bytes(size) # mean
          var   = self.read_bytes(size) # variance
          weights = norm_layer.set_weights([gamma, beta, mean, var])
        if len(conv_layer.get_weights()) > 1:
          bias   = self.read_bytes(np.prod(conv_layer.get_weights()[1].shape))
          kernel = self.read_bytes(np.prod(conv_layer.get_weights()[0].shape))
          kernel = kernel.reshape(list(reversed(conv_layer.get_weights()[0].sh
ape)))
          kernel = kernel.transpose([2,3,1,0])
          conv_layer.set_weights([kernel, bias])
        else:
          kernel = self.read_bytes(np.prod(conv_layer.get_weights()[0].shape))
```

```
          kernel = kernel.reshape(list(reversed(conv_layer.get_weights()[0].sh
ape)))
          kernel = kernel.transpose([2,3,1,0])
          conv_layer.set_weights([kernel])
      except ValueError:
        print("no convolution #" + str(i))

  def reset(self):
    self.offset = 0
def _conv_block(inp, convs, skip=True):
  x = inp
  count = 0
  for conv in convs:
    if count == (len(convs) - 2) and skip:
      skip_connection = x
    count += 1
    if conv['stride'] > 1: x = ZeroPadding2D(((1,0),(1,0)))(x) # peculiar padd
ing as darknet prefer left and top
    x = Conv2D(conv['filter'],
           conv['kernel'],
           strides=conv['stride'],
           padding='valid' if conv['stride'] > 1 else 'same', # peculiar paddi
ng as darknet prefer left and top
           name='conv_' + str(conv['layer_idx']),
           use_bias=False if conv['bnorm'] else True)(x)
    if conv['bnorm']: x = BatchNormalization(epsilon=0.001, name='bnorm_' + st
r(conv['layer_idx']))(x)
    if conv['leaky']: x = LeakyReLU(alpha=0.1, name='leaky_' + str(conv['layer
_idx']))(x)
  return add([skip_connection, x]) if skip else x

def make_yolov3_model():
  input_image = Input(shape=(None, None, 3))
  # Layer  0 => 4
  x = _conv_block(input_image, [{'filter': 32, 'kernel': 3, 'stride': 1, 'bnor
m': True, 'leaky': True, 'layer_idx': 0},
                  {'filter': 64, 'kernel': 3, 'stride': 2, 'bnorm': True, 'lea
ky': True, 'layer_idx': 1},
                  {'filter': 32, 'kernel': 1, 'stride': 1, 'bnorm': True, 'lea
ky': True, 'layer_idx': 2},
                  {'filter': 64, 'kernel': 3, 'stride': 1, 'bnorm': True, 'lea
ky': True, 'layer_idx': 3}])
  # Layer  5 => 8
  x = _conv_block(x, [{'filter': 128, 'kernel': 3, 'stride': 2, 'bnorm': True,
 'leaky': True, 'layer_idx': 5},
```

```
            {'filter':  64, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 6},
            {'filter': 128, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 7}])
  # Layer  9 => 11
  x = _conv_block(x, [{'filter':  64, 'kernel': 1, 'stride': 1, 'bnorm': True,
 'leaky': True, 'layer_idx': 9},
            {'filter': 128, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 10}])
  # Layer 12 => 15
  x = _conv_block(x, [{'filter': 256, 'kernel': 3, 'stride': 2, 'bnorm': True,
 'leaky': True, 'layer_idx': 12},
            {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 13},
            {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 14}])
  # Layer 16 => 36
  for i in range(7):
    x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': Tru
e, 'leaky': True, 'layer_idx': 16+i*3},
              {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky'
: True, 'layer_idx': 17+i*3}])
  skip_36 = x
  # Layer 37 => 40
  x = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 2, 'bnorm': True,
 'leaky': True, 'layer_idx': 37},
            {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 38},
            {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 39}])
  # Layer 41 => 61
  for i in range(7):
    x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': Tru
e, 'leaky': True, 'layer_idx': 41+i*3},
              {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky'
: True, 'layer_idx': 42+i*3}])
  skip_61 = x
  # Layer 62 => 65
  x = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 2, 'bnorm': True
, 'leaky': True, 'layer_idx': 62},
            {'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
 True, 'layer_idx': 63},
            {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
 True, 'layer_idx': 64}])
  # Layer 66 => 74
  for i in range(3):
```

```
    x = _conv_block(x, [{'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': Tr
ue, 'leaky': True, 'layer_idx': 66+i*3},
                {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky
': True, 'layer_idx': 67+i*3}])
  # Layer 75 => 79
  x = _conv_block(x, [{'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True
, 'leaky': True, 'layer_idx': 75},
            {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
 True, 'layer_idx': 76},
            {'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
 True, 'layer_idx': 77},
            {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
 True, 'layer_idx': 78},
            {'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
 True, 'layer_idx': 79}], skip=False)
  # Layer 80 => 82
  yolo_82 = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm'
: True,  'leaky': True,  'layer_idx': 80},
                {'filter':  255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'le
aky': False, 'layer_idx': 81}], skip=False)
  # Layer 83 => 86
  x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True,
 'leaky': True, 'layer_idx': 84}], skip=False)
  x = UpSampling2D(2)(x)
  x = concatenate([x, skip_61])
  # Layer 87 => 91
  x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True,
 'leaky': True, 'layer_idx': 87},
            {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 88},
            {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 89},
            {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 90},
            {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky':
True, 'layer_idx': 91}], skip=False)
  # Layer 92 => 94
  yolo_94 = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm':
 True,  'leaky': True,  'layer_idx': 92},
                {'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'lea
ky': False, 'layer_idx': 93}], skip=False)
  # Layer 95 => 98
  x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,
 'leaky': True,   'layer_idx': 96}], skip=False)
  x = UpSampling2D(2)(x)
  x = concatenate([x, skip_36])
```

```
  # Layer 99 => 106
  yolo_106 = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm'
: True,  'leaky': True,  'layer_idx': 99},
                {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True,  'le
aky': True,  'layer_idx': 100},
                {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,  'le
aky': True,  'layer_idx': 101},
                {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True,  'le
aky': True,  'layer_idx': 102},
                {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,  'le
aky': True,  'layer_idx': 103},
                {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True,  'le
aky': True,  'layer_idx': 104},
                {'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'le
aky': False, 'layer_idx': 105}], skip=False)
  model = Model(input_image, [yolo_82, yolo_94, yolo_106])
  return model
```

Load the Pre-trained person detector model and weights into memory

```
# define the yolo v3 model
yolov3 = make_yolov3_model()
# load the weights
weight_reader = WeightReader('/content/gdrive/My Drive/Pose_Estimation_Dataset
/yolov3.weights')

# set the weights
weight_reader.load_weights(yolov3)
```

Functions for refining results from person detector model and make prediction

```
# define the anchors
anchors = [[116,90, 156,198, 373,326], [30,61, 62,45, 59,119], [10,13, 16,30,
33,23]]

# define the probability threshold for detected objects
class_threshold = 0.75

# define the class labels
labels = ["person"]

class BoundBox:
  def __init__(self, xmin, ymin, xmax, ymax, objness = None, classes = None):
    self.xmin = xmin
    self.ymin = ymin
    self.xmax = xmax
    self.ymax = ymax
    self.objness = objness
    self.classes = classes
    self.label = -1
```

```
    self.score = -1

  def get_label(self):
    if self.label == -1:
      self.label = np.argmax(self.classes)

    return self.label

  def get_score(self):
    if self.score == -1:
      self.score = self.classes[self.get_label()]
    return self.get_score

def _sigmoid(x):
  return 1. /(1. + np.exp(-x))

def decode_netout(netout, anchors, obj_thresh, net_h, net_w):
  grid_h, grid_w = netout.shape[:2]
  nb_box = 3
  netout = netout.reshape((grid_h, grid_w, nb_box, -1))
  nb_class = netout.shape[-1] - 5
  boxes = []
  netout[..., :2]  = _sigmoid(netout[..., :2])
  netout[..., 4:]  = _sigmoid(netout[..., 4:])
  netout[..., 5:]  = netout[..., 4][..., np.newaxis] * netout[..., 5:]
  netout[..., 5:] *= netout[..., 5:] > obj_thresh

  for i in range(grid_h*grid_w):
    row = i / grid_w
    col = i % grid_w
    for b in range(nb_box):
      # 4th element is objectness score
      objectness = netout[int(row)][int(col)][b][4]
      if(objectness.all() <= obj_thresh): continue
      # first 4 elements are x, y, w, and h
      x, y, w, h = netout[int(row)][int(col)][b][:4]
      x = (col + x) / grid_w # center position, unit: image width
      y = (row + y) / grid_h # center position, unit: image height
      w = anchors[2 * b + 0] * np.exp(w) / net_w # unit: image width
      h = anchors[2 * b + 1] * np.exp(h) / net_h # unit: image height
      # last elements are class probabilities
      classes = netout[int(row)][col][b][5:]
      box = BoundBox(x-w/2, y-h/2, x+w/2, y+h/2, objectness, classes)
      boxes.append(box)
  return boxes
```

```
def correct_yolo_boxes(boxes, image_h, image_w, net_h, net_w):
  new_w, new_h = net_w, net_h
  for i in range(len(boxes)):
    x_offset, x_scale = (net_w - new_w)/2./net_w, float(new_w)/net_w
    y_offset, y_scale = (net_h - new_h)/2./net_h, float(new_h)/net_h
    boxes[i].xmin = int((boxes[i].xmin - x_offset) / x_scale * image_w)
    boxes[i].xmax = int((boxes[i].xmax - x_offset) / x_scale * image_w)
    boxes[i].ymin = int((boxes[i].ymin - y_offset) / y_scale * image_h)
    boxes[i].ymax = int((boxes[i].ymax - y_offset) / y_scale * image_h)

def _interval_overlap(interval_a, interval_b):
  x1, x2 = interval_a
  x3, x4 = interval_b
  if x3 < x1:
    if x4 < x1:
      return 0
    else:
      return min(x2,x4) - x1
  else:
    if x2 < x3:
       return 0
    else:
      return min(x2,x4) - x3

def bbox_iou(box1, box2):
  intersect_w = _interval_overlap([box1.xmin, box1.xmax], [box2.xmin, box2.xma
x])
  intersect_h = _interval_overlap([box1.ymin, box1.ymax], [box2.ymin, box2.yma
x])
  intersect = intersect_w * intersect_h
  w1, h1 = box1.xmax-box1.xmin, box1.ymax-box1.ymin
  w2, h2 = box2.xmax-box2.xmin, box2.ymax-box2.ymin
  union = w1*h1 + w2*h2 - intersect
  return float(intersect) / union

def do_nms(boxes, nms_thresh):
  if len(boxes) > 0:
    nb_class = len(boxes[0].classes)
  else:
    return
  for c in range(nb_class):
    sorted_indices = np.argsort([-box.classes[c] for box in boxes])
    for i in range(len(sorted_indices)):
      index_i = sorted_indices[i]
      if boxes[index_i].classes[c] == 0: continue
      for j in range(i+1, len(sorted_indices)):
```

```
        index_j = sorted_indices[j]
        if bbox_iou(boxes[index_i], boxes[index_j]) >= nms_thresh:
          boxes[index_j].classes[c] = 0

# get all of the results above a threshold
def get_boxes(boxes, labels, thresh):
  v_boxes, v_labels, v_scores = list(), list(), list()
  # enumerate all boxes
  for box in boxes:
    # enumerate all possible labels
    for i in range(len(labels)):
      # check if the threshold for this label is high enough
      if box.classes[i] > thresh:
        v_boxes.append(box)
        v_labels.append(labels[i])
        v_scores.append(box.classes[i]*100)
        # don't break, many labels may trigger for one box
  return v_boxes, v_labels, v_scores

# draw all results
def draw_boxes(filename, v_boxes, v_labels, v_scores):

  # load the image
  data = pyplot.imread(filename)
  data = cv2.resize(data, (224, 224))
  # plot the image
  pyplot.imshow(data)
  # get the context for drawing boxes
  ax = pyplot.gca()
  # plot each box
  for i in range(len(v_boxes)):
    box = v_boxes[i]
    # get coordinates
    y1, x1, y2, x2 = box.ymin, box.xmin, box.ymax, box.xmax
    # calculate width and height of the box
    width, height = x2 - x1, y2 - y1
    # create the shape
    rect = Rectangle((x1, y1), width, height, fill=False, color='yellow', line
width = '2')
    # draw the box
    ax.add_patch(rect)
    # draw text and score in top left corner
    label = "%s (%.3f)" % (v_labels[i], v_scores[i])
    pyplot.text(x1, y1, label, color='yellow')
  # show the plot
  pyplot.show()
```

```
def load_image_pixels(filename, shape):
  # load image to get its shape
  image = load_img(filename)
  image = image.resize((224,224))
  #width, height = image.size
  width, height = 224, 224
  # load image with required size
  image = load_img(filename, target_size=shape)
  image = img_to_array(image)

  # grayscale image normalization
  image = image.astype('float32')
  image /= 255.0

  # add a dimension so that we have one sample
  image = expand_dims(image, 0)
  return image, width, height
```

Posture estimation Network Architecture and Load training weights

```
def posture_estimation_model_func():

  # Instantiation
  Posture_Net = Sequential()

  #1st Convolutional Layer
  Posture_Net.add(Conv2D(filters=96, input_shape=(128,128,1), kernel_size=(11,
11), strides=(4,4), padding='same'))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))
  Posture_Net.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same')
)

  #2nd Convolutional Layer
  Posture_Net.add(Conv2D(filters=256, kernel_size=(5, 5), strides=(1,1), paddi
ng='same'))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))
  Posture_Net.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same')
)

  #3rd Convolutional Layer
  Posture_Net.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), paddin
g='same'))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))
```

```
  #4th Convolutional Layer
  Posture_Net.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), paddin
g='same'))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))

  #5th Convolutional Layer
  Posture_Net.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), paddin
g='same'))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))

  Posture_Net.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same')
)

  #Passing it to a Fully Connected layer
  Posture_Net.add(Flatten())
  # 1st Fully Connected Layer
  Posture_Net.add(Dense(4096))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))
  # Add Dropout to prevent overfitting
  Posture_Net.add(Dropout(0.4))

  #2nd Fully Connected Layer
  Posture_Net.add(Dense(4096))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))
  #Add Dropout
  Posture_Net.add(Dropout(0.4))

  #3rd Fully Connected Layer
  Posture_Net.add(Dense(1000))
  Posture_Net.add(BatchNormalization())
  Posture_Net.add(Activation('relu'))
  #Add Dropout
  Posture_Net.add(Dropout(0.4))

  #Output Layer
  Posture_Net.add(Dense(34))
  Posture_Net.add(BatchNormalization())

  #Model Summary
  #Posture_Net.summary()
  return Posture_Net
posture_estimation_model = posture_estimation_model_func()
```

```
posture_estimation_model.load_weights('/content/gdrive/My Drive/Pose_Estimation_Dataset/Model/weights_full_body_posture.hdf5')
```

Functions to calculate body posture score and grand score from individual body part score

```
def find_rula_posture_score_a(upper_arm_score, lower_arm_score, wrist_posture_score, wrist_twist_score):
  upper_posture_score = 0

  upper_arm_df = rula_group_a_df.loc[(rula_group_a_df[0] == upper_arm_score) | (rula_group_a_df[0] == 0)]

  lower_arm_df = upper_arm_df.loc[(rula_group_a_df[1] == lower_arm_score) | (rula_group_a_df[0] == 0)]

  lower_arm_df = lower_arm_df.reset_index(drop = True)
  for col in range(lower_arm_df.shape[1]):
    if lower_arm_df[col][0] == wrist_posture_score:
      if lower_arm_df[col][1] == wrist_twist_score:
        upper_posture_score = lower_arm_df[col][2]
  print('Score A: '+str(upper_posture_score))
  return upper_posture_score


def find_rula_posture_score_b(neck_posture_score, trunk_posture_score, leg_posture_score):
  lower_posture_score = 0
  neck_df = rula_group_b_df.loc[(rula_group_b_df[0] == neck_posture_score) | (rula_group_b_df[0] == 0)]

  neck_df = neck_df.reset_index(drop=True)
  for col in range(neck_df.shape[1]):
    if neck_df[col][0] == trunk_posture_score:
      if neck_df[col][1] == leg_posture_score:
        lower_posture_score = neck_df[col][2]

  print('Score B: '+str(lower_posture_score))
  return lower_posture_score


def find_rula_grand_score(upper_body_posture_score, lower_body_posture_score):
  # Assumption
  upper_body_posture_score = upper_body_posture_score+2
  lower_body_posture_score = lower_body_posture_score+2
  grand_score = 0
  for col in range(rula_grand_score_df.shape[1]):
    if rula_grand_score_df[col][0] == lower_body_posture_score:
      for row in range(rula_grand_score_df.shape[0]):
```

```
        if rula_grand_score_df[0][row] == upper_body_posture_score:
          grand_score = rula_grand_score_df[col][row]

  print('Grand score is: '+str(grand_score))
  return grand_score
```

Functions to calculate individual body part score from body angles

```
def calc_upper_arm_score(input_upper_arm_angle):
  print('Upper arm Flexion')
  if input_upper_arm_angle > 0:
    _is_upper_arm_flexion = True
    if 0 < input_upper_arm_angle <= 20:
      upper_arm_score = 1
    elif 20 < input_upper_arm_angle <= 45:
      upper_arm_score = 2
    elif 45 < input_upper_arm_angle <= 90:
      upper_arm_score = 3
    # Shoulder raised
    elif 90 <= input_upper_arm_angle:
      upper_arm_score = 4
  else:
    _is_upper_arm_flexion = True
    if -20 < input_upper_arm_angle <= 0:
      upper_arm_score = 1
    elif -45 < input_upper_arm_angle <= -20:
      upper_arm_score = 2
    elif -90 < input_upper_arm_angle <= -45:
      upper_arm_score = 3
    # Shoulder raised
    elif -90 <= input_upper_arm_angle:
      upper_arm_score = 4

  return _is_upper_arm_flexion, upper_arm_score

def calc_lower_arm_score(input_lower_arm_angle):
  lower_arm_score = 0

  if input_lower_arm_angle < 0:
    _is_upper_arm_flexion = False
    # lower_arm_score = 1
    print('Lower arm extension not possible')

  else:
    print('Lower arm flexion')
    _is_upper_arm_flexion = True
    if 60 < input_lower_arm_angle < 100:
      lower_arm_score = 1
```

```
    elif input_lower_arm_angle <= 60:
      lower_arm_score = 2
    elif input_lower_arm_angle >= 100:
      lower_arm_score = 2

  return _is_upper_arm_flexion, lower_arm_score

def calc_wrist_score(input_wrist_angle):

  if input_wrist_angle == 0:
    wrist_posture_score = 1
  elif 0<input_wrist_angle<=15:
    wrist_posture_score = 2
  elif input_wrist_angle > 15:
    wrist_posture_score = 3

  return wrist_posture_score

def calc_neck_score(input_neck_angle, input_neck_status):
  # Flexion or neutral
  if input_neck_status == True:
    if 0 <= input_neck_angle < 10:
      neck_posture_score = 1
    elif 10 <= input_neck_angle < 20:
      neck_posture_score = 2
    elif input_neck_angle >= 20:
      neck_posture_score = 3
  else:
    neck_posture_score = 4

  return neck_posture_score

def calc_neck_twist(left_shoulder_coordinate_pos, right_shoulder_coordinate_po
s,
                        nose_coordinate_pos, img_view):
  if img_view=='front':
    right_shoulder_joint_coordinate = np.array([right_shoulder_coordinate_pos[
0], right_shoulder_coordinate_pos[1]])
    left_shoulder_joint_coordinate = np.array([left_shoulder_coordinate_pos[0]
, left_shoulder_coordinate_pos[1]])
    nose_joint_coordinate = np.array([nose_coordinate_pos[0], left_shoulder_co
ordinate_pos[1]])

    dist_1 = int(math.sqrt(((nose_joint_coordinate[0]-
right_shoulder_joint_coordinate[0])**2)+((nose_joint_coordinate[1]-
right_shoulder_joint_coordinate[1])**2)))
```

```
    dist_2 = int(math.sqrt(((nose_joint_coordinate[0]-
left_shoulder_joint_coordinate[0])**2)+((nose_joint_coordinate[1]-
left_shoulder_joint_coordinate[1])**2)))

    print(dist_1, dist_2)

    if abs(dist_1-dist_2)>neck_twist_thres:
      _is_neck_side_twist_pos = True
    else:
      _is_neck_side_twist_pos = False
  else:
    print('Not an appropriate view. Needs front view')
    _is_neck_side_twist_pos = False

  return _is_neck_side_twist_pos

# distance between head and both shoulders must be same
def calc_neck_side_bending(trunk_coordinate_pos, neck_coordinate_pos, head_coo
rdinate_pos, img_view):
  if img_view == 'front':
    point_c = np.array([trunk_coordinate_pos[0], trunk_coordinate_pos[1]])
    base_point = np.array([neck_coordinate_pos[0], neck_coordinate_pos[1]])
    point_b = np.array([head_coordinate_pos[0], head_coordinate_pos[1]])

    ba = point_b - base_point
    bc = point_c - base_point

    neck_bend_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot
(ba, bc)))
    print('Func neck side bent angle: '+str(neck_bend_angle))

    neck_bend_angle = abs(neck_bend_angle)

    if neck_bend_angle<=180:
      neck_bend_angle = 180 - neck_bend_angle

    if neck_bend_angle>=5:
      _is_neck_side_bent_pos = True
    else:
      _is_neck_side_bent_pos = False
  else:
    print('Not an appropriate view. Needs front view')
    _is_neck_side_bent_pos = False

  return _is_neck_side_bent_pos
```

```
def calc_trunk_score(input_trunk_angle):
  print('Trunk flexion')
  if 0 < input_trunk_angle <= 20:
    trunk_posture_score = 2
  elif 20 < input_trunk_angle <= 60:
    trunk_posture_score = 3
  elif input_trunk_angle > 60:
    trunk_posture_score = 4
  elif input_trunk_angle==0:
    print('Neutral trunk position')
    trunk_posture_score = 1

  return trunk_posture_score

# difference of length of opposite pairs of trunk and shoulder must be less th
an 10
def calc_trunk_twist(left_shoulder_coordinate_pos, right_shoulder_coordinate_p
os,
                     left_trunk_coordinate_pos, right_trunk_coordinate_pos,
                     trunk_side_bend_status, view_img):

  if view_img=='front':
    right_shoulder_joint_coordinate = np.array([right_shoulder_coordinate_pos[
0], right_shoulder_coordinate_pos[1]])
    left_shoulder_joint_coordinate = np.array([left_shoulder_coordinate_pos[0]
, left_shoulder_coordinate_pos[1]])
    right_trunk_joint_coordinate = np.array([right_trunk_coordinate_pos[0], ri
ght_trunk_coordinate_pos[1]])
    left_trunk_joint_coordinate = np.array([left_trunk_coordinate_pos[0], left
_trunk_coordinate_pos[1]])

    dist_1 = int(math.sqrt(((left_trunk_joint_coordinate[0]-
right_shoulder_joint_coordinate[0])**2)+
                           ((left_trunk_joint_coordinate[1]-
right_shoulder_joint_coordinate[1])**2)))
    dist_2 = int(math.sqrt(((right_trunk_joint_coordinate[0]-
left_shoulder_joint_coordinate[0])**2)+
                           ((right_trunk_joint_coordinate[1]-
left_shoulder_joint_coordinate[1])**2)))

    print(dist_1, dist_2)

    if abs(dist_1-dist_2)>trunk_twist_thres:
      _is_trunk_side_twist_pos = True
      if trunk_side_bend_status:
```

```
        print('Trunk side twist result can not be relied because of side bent'
)
        if abs(dist_1-dist_2)>10:
          print('Strong probability of trunk twisting')
          _is_trunk_side_twist_pos = True
        else:
          _is_trunk_side_twist_pos = False
    else:
      _is_trunk_side_twist_pos = False

  else:
    print('Not an appropriate view. Needs front view')
    _is_trunk_side_twist_pos = False

  return _is_trunk_side_twist_pos

def calc_wrist_bent(elbow_coordinate_pos, wrist_coordinate_pos, knuckle_coordi
nate_pos, img_view):

  point_c = np.array([elbow_coordinate_pos[0], elbow_coordinate_pos[1]])
  base_point = np.array([wrist_coordinate_pos[0], wrist_coordinate_pos[1]])
  point_b = np.array([knuckle_coordinate_pos[0], knuckle_coordinate_pos[1]])
  ba = point_b - base_point
  bc = point_c - base_point

  wrist_bent_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot(
ba, bc)))
  print('Func wrist bent angle: '+str(wrist_bent_angle))

  wrist_bent_angle = abs(wrist_bent_angle)

  if wrist_bent_angle<=180:
    wrist_bent_angle = 180 - wrist_bent_angle

  wrist_bent_status = False
  print('Wrist bent angle: '+str(wrist_bent_angle))
  if img_view == 'front':
    if wrist_bent_angle>wrist_bend_thres:
      wrist_bent_status = True
    else:
      wrist_bent_status = False
  else:
    print('Not an appropriate view. Needs front view')
    wrist_bent_status = False
  return wrist_bent_status
```

```
def calc_trunk_side_bent(left_shoulder_coordinate_pos, right_shoulder_coordina
te_pos, img_view):
  if img_view=='front':

    right_shoulder_joint_coordinate = np.array([right_shoulder_coordinate_pos[
0], right_shoulder_coordinate_pos[1]])
    left_shoulder_joint_coordinate = np.array([left_shoulder_coordinate_pos[0]
, left_shoulder_coordinate_pos[1]])
    point_c = np.array([right_shoulder_coordinate_pos[0]*0.5, right_shoulder_c
oordinate_pos[1]])

    ba = left_shoulder_joint_coordinate - right_shoulder_joint_coordinate
    bc = point_c - right_shoulder_joint_coordinate

    trunk_side_bent_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]),
np.dot(ba, bc)))
    print('Func trunk side bend angle: '+str(trunk_side_bent_angle))

    trunk_side_bent_angle = 180  - abs(trunk_side_bent_angle)

    if trunk_side_bent_angle>=trunk_side_bend_thres:
      is_trunk_side_bent_pos = True
    else:
      is_trunk_side_bent_pos = False

  else:
    print('Not an appropriate view. Needs front view')
    is_trunk_side_bent_pos = False
  print('Trunk side bend angle: '+str(trunk_side_bent_angle))
  return is_trunk_side_bent_pos

# difference of length of right and left legs(ankle to knee) must be less than
 10
def calc_leg_support(left_ankle_coordinate_pos, right_ankle_coordinate_pos):

  right_ankle_joint_coordinate = np.array([right_ankle_coordinate_pos[0], righ
t_ankle_coordinate_pos[1]])
  left_ankle_joint_coordinate = np.array([left_ankle_coordinate_pos[0], left_a
nkle_coordinate_pos[1]])

  dist_1 = left_ankle_joint_coordinate[1]
  dist_2 = right_ankle_joint_coordinate[1]
  print(dist_1, dist_2)
  if abs(dist_1-dist_2)>10:
    _is_leg_supported_pos = False
  else:
```

```
    _is_leg_supported_pos = True

  return _is_leg_supported_pos
# trunk,shoulder & shoulder,Elbow
# point_a[397,255.5], point_b[502,267], base_point[502.5,166.0]
def calc_upper_arm_angle(point_a, base_point, point_b):

  hip_joint_coordinate = np.array([point_a[0], point_a[1]])
  shoulder_joint_coordinate = np.array([base_point[0], base_point[1]])
  elbow_joint_coordinate = np.array([point_b[0], point_b[1]])

  ba = hip_joint_coordinate - shoulder_joint_coordinate
  bc = elbow_joint_coordinate - shoulder_joint_coordinate

  # positive if elbow is behind shoulder (Anti-clockwise) i.e extension
  upper_arm_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot(b
a, bc)))

  return upper_arm_angle


# Extend the line connecting shoulder and elbow and find angle between upper a
rm and lower arm(elbow, wrist)
# Shoulder[502.5,166.0]; elbow[502, 267]; wrist[542.5,345.5]
def calc_lower_arm_angle(input_shoulder_joint_coordinate, input_elbow_joint_co
ordinate, input_wrist_joint_coordinate):

  point_c = np.array([input_shoulder_joint_coordinate[0], input_shoulder_joint
_coordinate[1]])
  base_point = np.array([input_elbow_joint_coordinate[0], input_elbow_joint_co
ordinate[1]])
  point_b = np.array([input_wrist_joint_coordinate[0], input_wrist_joint_coord
inate[1]])

  ba = point_b - base_point
  bc = point_c - base_point

  lower_arm_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot(b
a, bc)))
  print('Func lower arm angle: '+str(lower_arm_angle))
  lower_arm_angle = 180 - abs(lower_arm_angle)

  return lower_arm_angle

def calc_lower_arm_work_midline(right_shoulder_joint_coordinate, left_shoulder
_joint_coordinate,
```

```
                                    right_wrist_coordinate, left_wrist_coordinate,
 img_view):
    if img_view == 'front':
      if right_shoulder_joint_coordinate[0] - left_shoulder_joint_coordinate[0
] > 0:
        print('front view with left arm towards left side of image')
        if left_wrist_coordinate[0] > right_wrist_coordinate[0]:
          is_lower_arm_working_midline = False
        else:
          is_lower_arm_working_midline = True

      elif left_shoulder_joint_coordinate[0] - right_shoulder_joint_coordinate
[0] > 0:
        print('front view with right arm towards left side of image')
        if right_wrist_coordinate[0] > left_wrist_coordinate[0]:
          is_lower_arm_working_midline = True
        else:
          is_lower_arm_working_midline = False
    # since it's not front view it can not be computed
    elif img_view == 'left':
      print('a left view image')
      if (left_wrist_coordinate[0] > right_wrist_coordinate[0]) and (left_wris
t_coordinate[1] <= right_wrist_coordinate[1]):
        is_lower_arm_working_midline = True
      else:
        is_lower_arm_working_midline = False
    elif img_view == 'right':
      print('a right view image')
      if (right_wrist_coordinate[0] > left_wrist_coordinate[0]) and (right_wri
st_coordinate[1] <= left_wrist_coordinate[1]):
        is_lower_arm_working_midline = True
      else:
        is_lower_arm_working_midline = False

    return is_lower_arm_working_midline

# line between elbow and wrist extended and interior angle is calculated betwe
en lower arm and wrist(wrist, knuckle)
def calc_wrist_posture_angle(elbow_joint_coordinate, wrist_joint_coordinate, k
nuckle_joint_coordinate):
    point_c = np.array([elbow_joint_coordinate[0], elbow_joint_coordinate[1]])
    base_point = np.array([wrist_joint_coordinate[0], wrist_joint_coordinate[1
]])
    point_b = np.array([knuckle_joint_coordinate[0], knuckle_joint_coordinate[
1]])
    ba = point_b - base_point
```

```
    bc = point_c - base_point

    wrist_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot(ba,
 bc)))
    print('Func wrist angle: '+str(wrist_angle))

    if wrist_angle<0:
      # clockwise
      print('wrist flexion')
      wrist_status = True
    elif wrist_angle>0:
      # anticlockwise
      print('wrist extension')
      wrist_status = False
    else:
      # in same line
      print('neutral')
      wrist_status = True

    wrist_angle = abs(wrist_angle)

    if wrist_angle<=180:
      wrist_angle = 180 - wrist_angle

    return wrist_angle

# [397,255.5], [533, 73.5], [639.5, 6.5]
def calc_neck_posture_angle(trunk_joint_coordinate, neck_joint_coordinate, hea
d_joint_coordinate, image_view):
    point_c = np.array([trunk_joint_coordinate[0], trunk_joint_coordinate[1]])
    base_point = np.array([neck_joint_coordinate[0], neck_joint_coordinate[1]]
)
    point_b = np.array([head_joint_coordinate[0], head_joint_coordinate[1]])

    ba = point_b - base_point
    bc = point_c - base_point

    # positive if head is behind neck i.e extension
    neck_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot(ba,
bc)))
    print('Func neck angle: '+str(neck_angle))
    print('View of the image: '+str(image_view))
    if image_view =='left':
      print('left view')
      if neck_angle<0:
        neck_status = True
```

```
        print('Neck flexion')
      elif neck_angle>0:
        neck_status = False
        print('Neck extension')
      else:
        neck_status = True
        print('Neck neutral')
    elif image_view =='right':
      print('right view')
      if neck_angle>0:
        neck_status = True
        print('Neck flexion')
      elif neck_angle<0:
        neck_status = False
        print('Neck extension')
      else:
        neck_status = True
        print('Neck neutral')


    neck_angle = abs(neck_angle)

    if neck_angle<=180:
      neck_angle = 180 - neck_angle

    return neck_status, neck_angle

# [395, 428.5], [370.5, 247.5], [486, 68.5]
def calc_trunk_posture_angle(vertical_joint_coordinate, trunk_joint_coordinate
, neck_joint_coordinate, image_view):
  point_b = np.array([vertical_joint_coordinate[0], vertical_joint_coordinate[
1]])
  base_point = np.array([trunk_joint_coordinate[0], trunk_joint_coordinate[1]]
)
  point_c = np.array([neck_joint_coordinate[0], neck_joint_coordinate[1]])

  ba = point_b - base_point
  bc = point_c - base_point

  trunk_angle = np.degrees(np.math.atan2(np.linalg.det([ba, bc]), np.dot(ba, b
c)))

  print('Func trunk angle: '+str(trunk_angle))
  if image_view=='left':
    print('left view image')
  elif image_view=='right':
```

```
    print('right view image')
  else:
    print('front view image')

  trunk_angle = abs(trunk_angle)

  return trunk_angle
```

Functions to decode outputs of posture estimation network

```
def decode_pose_estimate_net(input_test_img_name_s1, input_extracted_test_body
_s1, model_name):
      input_test_label_data_list_s1 = []
      pre_processed_test_image_s1 = pre_process_body_image(input_extracted_tes
t_body_s1[0], mode='test')
      body_bbox_s1_x1 = input_extracted_test_body_s1[1][0][0]
      body_bbox_s1_y1 = input_extracted_test_body_s1[1][0][1]
      body_bbox_s1_x2 = input_extracted_test_body_s1[1][0][2]
      body_bbox_s1_y2 = input_extracted_test_body_s1[1][0][3]

      x_test_s1 = np.array(pre_processed_test_image_s1)
      x_test_s1 = np.array(x_test_s1).reshape([-
1, body_model_img_size_width, body_model_img_size_height, body_channel])
      print(x_test_s1.shape)

      # predict keypoints in image
      prediction_s1 = predict_model(model_name, x_test_s1)

      test_src_img_s1 = cv2.imread(input_test_img_name_s1,cv2.IMREAD_COLOR)
      test_src_img_s1 = cv2.resize(test_src_img_s1, (224, 224))
      test_img_s1_height, test_img_s1_width, test_img_channel = test_src_img_s
1.shape


      body_bbox_s1_width = int(abs(body_bbox_s1_x1 - body_bbox_s1_x2))
      body_bbox_s1_height = int(abs(body_bbox_s1_y1 - body_bbox_s1_y2))

      body_bbox_center_x1 = int(body_bbox_s1_x1+(body_bbox_s1_width*0.5))
      body_bbox_center_y1 = int(body_bbox_s1_y1+(body_bbox_s1_height*0.5))
      # calculate absolute coordinates of labels
      for test_i in range(0, classes_count):
        test_img_label_s1 = ''
        test_s1_x_coordinate =  int(body_bbox_s1_x1+((prediction_s1[0][2*test_
i]*(224/128))*(body_bbox_s1_width/224)))
        test_s1_y_coordinate = int(body_bbox_s1_y1 +((prediction_s1[0][(2*test
_i)+1]*(224/128))*(body_bbox_s1_height/224)))
```

```
        for test_class_n, test_class_id in model_class_id_list.items():
          if test_class_id==test_i+1:
            test_img_label_s1 = test_class_n
        input_test_label_data_list_s1.append([test_s1_x_coordinate, test_s1_y_
coordinate, test_img_label_s1])

      # calculate neck coordinate
      input_test_label_data_list_s1.append([abs(input_test_label_data_list_s1[
0][0]+input_test_label_data_list_s1[1][0])*0.5,
                                            abs(input_test_label_data_list_s1[
0][1]+input_test_label_data_list_s1[1][1])*0.50,
                                            'neck'])
      head_y = abs(input_test_label_data_list_s1[17][1] - input_test_label_dat
a_list_s1[0][1])
      input_test_label_data_list_s1.append([abs(input_test_label_data_list_s1[
14][0]+input_test_label_data_list_s1[15][0])*0.5,
                                            (abs(input_test_label_data_list_s1
[14][1]+input_test_label_data_list_s1[15][1])*0.5)-head_y,
                                            'head'])

      # Plot keypoints
      for test_label_data in input_test_label_data_list_s1:
        print('Plotting point '+str(int(test_label_data[0])) + str(', ' )+ str
(int(test_label_data[1])) + ' for label '+ str(test_label_data[2]))
        test_src_img_s1 = cv2.circle(img=test_src_img_s1, center=(int(test_lab
el_data[0]), int(test_label_data[1])), radius=2,
                                     color=(255, 0, 0), thickness=-
1)
      cv2_imshow(test_src_img_s1)
      return input_test_label_data_list_s1
def plot_estimate_rula_score(input_test_front_img, input_test_side_img, img_vi
ew):
  # Fetch the side and front view image
  test_img_front = input_test_front_img
  test_img_side = input_test_side_img
  if img_view=='R':
    view = 'right'
  elif img_view=='L':
    view = 'left'
  extracted_test_bodies_side = extract_person_from_image(test_img_side)

  if len(extracted_test_bodies_side)>0:
    print('No of people detected: '+str(len(extracted_test_bodies_side)))
    for extracted_test_body_side in extracted_test_bodies_side:
      test_label_data_list_side=[]
```

```
      test_label_data_list_side = decode_pose_estimate_net(test_img_side,extra
cted_test_body_side, posture_estimation_model)
      # continue
      body_keypoint_df = pd.DataFrame(test_label_data_list_side)
      body_keypoint_df = body_keypoint_df.set_index([2])
      left_shoulder_keypoint_df = body_keypoint_df.loc['left_shoulder']
      left_shoulder_keypoint_coordinates = [left_shoulder_keypoint_df[0], left
_shoulder_keypoint_df[1]]
      right_shoulder_keypoint_df = body_keypoint_df.loc['right_shoulder']
      right_shoulder_keypoint_coordinates = [right_shoulder_keypoint_df[0], ri
ght_shoulder_keypoint_df[1]]

      mid_shoulder_keypoint_coordinates = [abs(left_shoulder_keypoint_df[0]+ri
ght_shoulder_keypoint_df[0])*0.5,
                                           abs(left_shoulder_keypoint_df[1]+right
_shoulder_keypoint_df[1])*0.5]

      neck_keypoint_df = body_keypoint_df.loc['neck']
      neck_keypoint_coordinates = [neck_keypoint_df[0], neck_keypoint_df[1]]
      nose_keypoint_df = body_keypoint_df.loc['nose']
      nose_keypoint_coordinates = [nose_keypoint_df[0], nose_keypoint_df[1]]

      head_keypoint_df = body_keypoint_df.loc['head']
      head_keypoint_coordinates = [head_keypoint_df[0], head_keypoint_df[1]]
      left_elbow_keypoint_df = body_keypoint_df.loc['left_elbow']
      left_elbow_keypoint_coordinates = [left_elbow_keypoint_df[0], left_elbow
_keypoint_df[1]]
      right_elbow_keypoint_df = body_keypoint_df.loc['right_elbow']
      right_elbow_keypoint_coordinates = [right_elbow_keypoint_df[0], right_el
bow_keypoint_df[1]]
      left_wrist_keypoint_df = body_keypoint_df.loc['left_wrist']
      left_wrist_keypoint_coordinates = [left_wrist_keypoint_df[0], left_wrist
_keypoint_df[1]]
      right_wrist_keypoint_df = body_keypoint_df.loc['right_wrist']
      right_wrist_keypoint_coordinates = [right_wrist_keypoint_df[0], right_wr
ist_keypoint_df[1]]
      left_knuckle_keypoint_df = body_keypoint_df.loc['left_knuckle']
      left_knuckle_keypoint_coordinates = [left_knuckle_keypoint_df[0], left_k
nuckle_keypoint_df[1]]
      right_knuckle_keypoint_df = body_keypoint_df.loc['right_knuckle']
      right_knuckle_keypoint_coordinates = [right_knuckle_keypoint_df[0], righ
t_knuckle_keypoint_df[1]]

      left_trunk_keypoint_df = body_keypoint_df.loc['left_trunk']
      left_trunk_keypoint_coordinates = [left_trunk_keypoint_df[0], left_trunk
_keypoint_df[1]]
```

```
      right_trunk_keypoint_df = body_keypoint_df.loc['right_trunk']
      right_trunk_keypoint_coordinates = [right_trunk_keypoint_df[0], right_tr
unk_keypoint_df[1]]

      mid_trunk_keypoint_coordinates = [abs(left_trunk_keypoint_df[0]+right_tr
unk_keypoint_df[0])*0.5,
                                        abs(left_trunk_keypoint_df[1]+right_tr
unk_keypoint_df[1])*0.5]

      mid_vertical_keypoint_coordinates = [abs(left_trunk_keypoint_df[0]+right
_trunk_keypoint_df[0])*0.5,
                                           nose_keypoint_df[1]]

      # Upper Arm Angle
      img_left_upper_arm_angle = calc_upper_arm_angle(left_trunk_keypoint_coor
dinates,
                                                      left_shoulder_keypoint_c
oordinates, left_elbow_keypoint_coordinates)
      print('Left Upper arm angle: '+str(img_left_upper_arm_angle))
      # status: true means flexion
      img_left_upper_arm_status, img_left_upper_arm_score = calc_upper_arm_sco
re(img_left_upper_arm_angle)
      print('Corresponding Score: '+ str(img_left_upper_arm_score))
      img_right_upper_arm_angle = calc_upper_arm_angle(right_trunk_keypoint_co
ordinates,
                                                       right_shoulder_keypoint_
coordinates, right_elbow_keypoint_coordinates)
      print('Right Upper arm angle: '+str(img_right_upper_arm_angle))


      img_right_upper_arm_status, img_right_upper_arm_score = calc_upper_arm_s
core(img_right_upper_arm_angle)
      print('Corresponding Score: '+ str(img_right_upper_arm_score))

      upper_body_lean_angle = calc_trunk_posture_angle(mid_vertical_keypoint_c
oordinates, mid_trunk_keypoint_coordinates,
                                                       mid_shoulder_keypoint_coordina
tes, view)
      print('Upper body lean angle: ' +str(upper_body_lean_angle))

      if 20<=upper_body_lean_angle<=70 and img_left_upper_arm_status and is_re
sted:
        _is_operator_lean = True
        print('Operator leaned')
        img_left_upper_arm_score = img_left_upper_arm_score - 1
        img_right_upper_arm_score = img_right_upper_arm_score - 1
```

```
        print('Upper arm score updated due to lean posture: '+str(img_left_upp
er_arm_score))
      else:
        _is_operator_lean = False
        print('Operator not leaned')

      # Lower Arm Posture
      img_left_lower_arm_angle = calc_lower_arm_angle(left_shoulder_keypoint_c
oordinates,
                                                      left_elbow_keypoint_coor
dinates, left_wrist_keypoint_coordinates)
      print('Left Lower arm angle: '+str(img_left_lower_arm_angle))
      left_lower_arm_status, img_left_lower_arm_score = calc_lower_arm_score(i
mg_left_lower_arm_angle)
      print('Corresponding Score: '+ str(img_left_lower_arm_score))
      img_right_lower_arm_angle = calc_lower_arm_angle(right_shoulder_keypoint
_coordinates,
                                                       right_elbow_keypoint_coo
rdinates, right_wrist_keypoint_coordinates)
      print('Right Lower arm angle: '+str(img_right_lower_arm_angle))
      right_lower_arm_status, img_right_lower_arm_score = calc_lower_arm_score
(img_right_lower_arm_angle)
      print('Corresponding Score: '+ str(img_right_lower_arm_score))

      print('Left wrist posture')
      img_left_wrist_angle = calc_wrist_posture_angle(left_elbow_keypoint_coor
dinates,
                                                      left_wrist_keypoint_coor
dinates, left_knuckle_keypoint_coordinates)
      print('Left wrist angle: '+str(img_left_wrist_angle))
      img_left_wrist_score = calc_wrist_score(img_left_wrist_angle)
      print('Corresponding Score: '+ str(img_left_wrist_score))

      print('Right wrist posture')
      img_right_wrist_angle = calc_wrist_posture_angle(right_elbow_keypoint_co
ordinates,
                                                       right_wrist_keypoint_coo
rdinates, right_knuckle_keypoint_coordinates)
      print('Right wrist angle: '+str(img_right_wrist_angle))
      img_right_wrist_score = calc_wrist_score(img_right_wrist_angle)
      print('Corresponding Score: '+ str(img_right_wrist_score))

      is_left_wrist_twist = False
      if is_left_wrist_twist:
        img_left_wrist_twist_score = 2
      else:
```

```
        img_left_wrist_twist_score = 1

      print('Left Wrist Twist Score: '+ str(img_left_wrist_twist_score))

      is_right_wrist_twist = False
      if is_right_wrist_twist:
        img_right_wrist_twist_score = 2
      else:
        img_right_wrist_twist_score = 1

      print('Right Wrist Twist Score: '+ str(img_right_wrist_twist_score))

      # Neck Posture
      if nose_keypoint_coordinates[0]>0:
        img_neck_status, img_neck_angle = calc_neck_posture_angle(mid_trunk_ke
ypoint_coordinates, neck_keypoint_coordinates,
                                                  head_keypoint_coordinates, vie
w)
      else:
        print('Nose coordinates are unavailable probably because face is undet
ected')
        img_neck_angle = 0
        img_neck_status = True

      print('Neck angle: '+str(img_neck_angle))
      print('Neck Flexion: '+str(img_neck_status))
      img_neck_score = calc_neck_score(img_neck_angle, img_neck_status)
      print('Corresponding Score: '+ str(img_neck_score))

      # Trunk Posture
      img_trunk_angle = calc_trunk_posture_angle(mid_vertical_keypoint_coordin
ates, mid_trunk_keypoint_coordinates,
                                                  neck_keypoint_coordinates, vie
w)
      print('trunk angle: '+str(img_trunk_angle))
      img_trunk_score = calc_trunk_score(img_trunk_angle)
      print('Corresponding Score: '+ str(img_trunk_score))

      # Side body calculation ends
      break
  # Front body starts
  extracted_test_bodies_front = extract_person_from_image(test_img_front)

  if len(extracted_test_bodies_front)>0:
    print('No of people detected: '+str(len(extracted_test_bodies_front)))
    for extracted_test_body_front in extracted_test_bodies_front:
```

```
      test_label_data_list_front=[]
      test_label_data_list_front = decode_pose_estimate_net(test_img_front,ext
racted_test_body_front, posture_estimation_model)
      # continue
      body_keypoint_df = pd.DataFrame(test_label_data_list_front)
      body_keypoint_df = body_keypoint_df.set_index([2])
      left_shoulder_keypoint_df = body_keypoint_df.loc['left_shoulder']
      left_shoulder_keypoint_coordinates = [left_shoulder_keypoint_df[0], left
_shoulder_keypoint_df[1]]
      right_shoulder_keypoint_df = body_keypoint_df.loc['right_shoulder']
      right_shoulder_keypoint_coordinates = [right_shoulder_keypoint_df[0], ri
ght_shoulder_keypoint_df[1]]

      mid_shoulder_keypoint_coordinates = [abs(left_shoulder_keypoint_df[0]+ri
ght_shoulder_keypoint_df[0])*0.5,
                                           abs(left_shoulder_keypoint_df[1]+right
_shoulder_keypoint_df[1])*0.5]

      neck_keypoint_df = body_keypoint_df.loc['neck']
      neck_keypoint_coordinates = [neck_keypoint_df[0], neck_keypoint_df[1]]
      nose_keypoint_df = body_keypoint_df.loc['nose']
      nose_keypoint_coordinates = [nose_keypoint_df[0], nose_keypoint_df[1]]
      right_eye_keypoint_df = body_keypoint_df.loc['right_eye']
      right_eye_keypoint_coordinates = [right_eye_keypoint_df[0], right_eye_ke
ypoint_df[1]]
      left_eye_keypoint_df = body_keypoint_df.loc['left_eye']
      left_eye_keypoint_coordinates = [left_eye_keypoint_df[0], left_eye_keypo
int_df[1]]
      head_keypoint_df = body_keypoint_df.loc['head']
      head_keypoint_coordinates = [head_keypoint_df[0], head_keypoint_df[1]]
      left_elbow_keypoint_df = body_keypoint_df.loc['left_elbow']
      left_elbow_keypoint_coordinates = [left_elbow_keypoint_df[0], left_elbow
_keypoint_df[1]]
      right_elbow_keypoint_df = body_keypoint_df.loc['right_elbow']
      right_elbow_keypoint_coordinates = [right_elbow_keypoint_df[0], right_el
bow_keypoint_df[1]]
      left_wrist_keypoint_df = body_keypoint_df.loc['left_wrist']
      left_wrist_keypoint_coordinates = [left_wrist_keypoint_df[0], left_wrist
_keypoint_df[1]]
      right_wrist_keypoint_df = body_keypoint_df.loc['right_wrist']
      right_wrist_keypoint_coordinates = [right_wrist_keypoint_df[0], right_wr
ist_keypoint_df[1]]
      left_knuckle_keypoint_df = body_keypoint_df.loc['left_knuckle']
      left_knuckle_keypoint_coordinates = [left_knuckle_keypoint_df[0], left_k
nuckle_keypoint_df[1]]
      right_knuckle_keypoint_df = body_keypoint_df.loc['right_knuckle']
```

```
      right_knuckle_keypoint_coordinates = [right_knuckle_keypoint_df[0], righ
t_knuckle_keypoint_df[1]]

      left_trunk_keypoint_df = body_keypoint_df.loc['left_trunk']
      left_trunk_keypoint_coordinates = [left_trunk_keypoint_df[0], left_trunk
_keypoint_df[1]]
      right_trunk_keypoint_df = body_keypoint_df.loc['right_trunk']
      right_trunk_keypoint_coordinates = [right_trunk_keypoint_df[0], right_tr
unk_keypoint_df[1]]

      mid_trunk_keypoint_coordinates = [abs(left_trunk_keypoint_df[0]+right_tr
unk_keypoint_df[0])*0.5,
                                        abs(left_trunk_keypoint_df[1]+right_tr
unk_keypoint_df[1])*0.5]

      left_knee_keypoint_df = body_keypoint_df.loc['left_knee']
      left_knee_keypoint_coordinates = [left_knee_keypoint_df[0], left_knee_ke
ypoint_df[1]]
      right_knee_keypoint_df = body_keypoint_df.loc['right_knee']
      right_knee_keypoint_coordinates = [right_knee_keypoint_df[0], right_knee
_keypoint_df[1]]

      mid_vertical_keypoint_coordinates = [abs(left_trunk_keypoint_df[0]+right
_trunk_keypoint_df[0])*0.5,
                                           nose_keypoint_df[1]]

      left_ankle_keypoint_df = body_keypoint_df.loc['left_ankle']
      left_ankle_keypoint_coordinates = [left_ankle_keypoint_df[0],left_ankle_
keypoint_df[1]]
      right_ankle_keypoint_df = body_keypoint_df.loc['right_ankle']
      right_ankle_keypoint_coordinates = [right_ankle_keypoint_df[0],right_ank
le_keypoint_df[1]]

      if (left_shoulder_keypoint_coordinates[1]-
left_elbow_keypoint_coordinates[1])>=-10:
        print(left_shoulder_keypoint_coordinates[1], left_elbow_keypoint_coord
inates[1])
        _is_left_shoulder_abducted = True
      else:
        _is_left_shoulder_abducted = False

      if (right_shoulder_keypoint_coordinates[1]-
right_elbow_keypoint_coordinates[1])>=-10:
        print(right_shoulder_keypoint_coordinates[1],right_elbow_keypoint_coor
dinates[1])
        _is_right_shoulder_abducted = True
```

```
      else:
        _is_right_shoulder_abducted = False

      print('Is Left Shoulder Abducted: '+str(_is_left_shoulder_abducted))
      if _is_left_shoulder_abducted:
        img_left_upper_arm_score = img_left_upper_arm_score + 1
        print('left upper arm score updated due to abduction: '+str(img_left_u
pper_arm_score))


      print('Is Right Shoulder Abducted: '+str(_is_right_shoulder_abducted))
      if _is_right_shoulder_abducted:
        img_right_upper_arm_score = img_right_upper_arm_score + 1
        print('right upper arm score updated due to abduction: '+str(img_right
_upper_arm_score))


      # Shoulder is elevated if upper arm angle>90*
      _is_left_shoulder_raised = False if abs(img_left_upper_arm_angle)>=90 el
se False
      print('Is left Shoulder raised: '+str(_is_left_shoulder_raised))
      if _is_left_shoulder_raised:
        img_left_upper_arm_score = img_left_upper_arm_score + 1
        print('left upper arm score updated due to shoulder raise: '+str(img_l
eft_upper_arm_score))

      _is_right_shoulder_raised = False if abs(img_right_upper_arm_angle)>=90
else False
      print('Is right Shoulder raised: '+str(_is_right_shoulder_raised))
      if _is_right_shoulder_raised:
        img_right_upper_arm_score = img_right_upper_arm_score + 1
        print('right upper arm score updated due to shoulder raise: '+str(img_
right_upper_arm_score))


      if img_right_upper_arm_score==0:
        print('Right Upper arm score was 1 and made 0 because posture is leane
d and rested')
        img_right_upper_arm_score = 1
      if img_left_upper_arm_score==0:
        print('Left Upper arm score was 1 and made 0 because posture is leaned
 and rested')
        img_left_upper_arm_score = 1

      print('Final left upper arm score: '+str(img_left_upper_arm_score))
      print('Final right upper arm score: '+str(img_right_upper_arm_score))
```

```
      img_lower_arm_working_midline = calc_lower_arm_work_midline(right_should
er_keypoint_coordinates,
                                                                  left_shoulde
r_keypoint_coordinates,
                                                                  right_ankle_
keypoint_coordinates,
                                                                  left_ankle_k
eypoint_coordinates,
                                                                  'front')
      print('is lower arm working midline '+str(img_lower_arm_working_midline)
)
      if img_lower_arm_working_midline:
        print('Lower arm score updated due to arm working midline posture')
        img_left_lower_arm_score = img_left_lower_arm_score + 1
        img_right_lower_arm_score = img_right_lower_arm_score + 1

      print('Final left lower arm score: '+str(img_left_lower_arm_score))
      print('Final right lower arm score: '+str(img_right_lower_arm_score))

      # is_left_wrist_bent_away_midline = calc_wrist_bent(left_elbow_keypoint_
coordinates, left_wrist_keypoint_coordinates,
      #                                                   left_knuckle_keypoin
t_coordinates,'front')
      is_left_wrist_bent_away_midline = False

      print('Is left wrist bent away from mid: '+str(is_left_wrist_bent_away_m
idline))
      if is_left_wrist_bent_away_midline:
        img_left_wrist_score = img_left_wrist_score+1
        print('Left Wrist Score updated because wrist bent away midline: '+ st
r(img_left_wrist_score))

      # is_right_wrist_bent_away_midline = calc_wrist_bent(right_elbow_keypoin
t_coordinates,right_wrist_keypoint_coordinates,
      #                                                    right_knuckle_keypo
int_coordinates,'front')
      is_right_wrist_bent_away_midline=False
      print('Is right wrist bent away from mid: '+str(is_right_wrist_bent_away
_midline))
      if is_right_wrist_bent_away_midline:
        img_right_wrist_score = img_right_wrist_score+1
        print('Right Wrist Score updated because wrist bent away midline: '+ s
tr(img_right_wrist_score))

      if nose_keypoint_coordinates[0]>0:
```

```
        img_neck_twist = calc_neck_twist(left_shoulder_keypoint_coordinates, r
ight_shoulder_keypoint_coordinates,
                                         nose_keypoint_coordinates, 'front')
        img_neck_side_bending = calc_neck_side_bending(mid_trunk_keypoint_coor
dinates, neck_keypoint_coordinates,
                                                       head_keypoint_coordinate
s, 'front')
      else:
        print('Nose coordinates are unavailable probably because face is undet
ected')
        img_neck_twist = False
        img_neck_side_bending = False

      print('is neck twisted: '+str(img_neck_twist))
      if img_neck_twist:
        img_neck_score = img_neck_score + 1
        print('Neck Score updated due to neck twisting: '+ str(img_neck_score)
)

      print('is neck side bending: '+str(img_neck_side_bending))
      if img_neck_side_bending:
        img_neck_score = img_neck_score + 1
        print('Neck Score updated due to side bending: '+ str(img_neck_score))

      img_trunk_side_bending = calc_trunk_side_bent(left_shoulder_keypoint_coo
rdinates, right_shoulder_keypoint_coordinates, 'front')
      print('is trunk side bent: '+str(img_trunk_side_bending))

      if img_trunk_side_bending:
        img_trunk_score = img_trunk_score + 1
        print('Trunk Score updated due to side bending: '+ str(img_trunk_score
))

      img_trunk_twist = calc_trunk_twist(left_shoulder_keypoint_coordinates, r
ight_shoulder_keypoint_coordinates,
                                         left_trunk_keypoint_coordinates, right
_trunk_keypoint_coordinates, img_trunk_side_bending, 'front')
      print('is trunk twist: '+str(img_trunk_twist))
      if img_trunk_twist:
        img_trunk_score = img_trunk_score + 1
        print('Trunk Score updated due to twisting: '+ str(img_trunk_score))

      img_leg_support_status = calc_leg_support(left_ankle_keypoint_coordinate
s, right_ankle_keypoint_coordinates)
      print('is leg supported: '+str(img_leg_support_status))
      if img_leg_support_status:
```

```
        img_leg_posture_score = 1
      else:
        img_leg_posture_score = 2
      print('Corresponding Score: '+ str(img_leg_posture_score))

      # front view calculation ends
      break

    # Main Rula score estimation starts
    print('Finding posture score for left side group A')
    print(img_left_upper_arm_score, img_left_lower_arm_score, img_left_wrist_s
core, img_left_wrist_twist_score)

    left_posture_score_a = find_rula_posture_score_a(upper_arm_score=img_left_
upper_arm_score,
                                                     lower_arm_score=img_left_l
ower_arm_score,
                                                     wrist_posture_score=img_le
ft_wrist_score,
                                                     wrist_twist_score=img_left
_wrist_twist_score)

    print('Finding posture score for right side group A')
    print(img_right_upper_arm_score, img_left_lower_arm_score, img_right_wrist
_score, img_right_wrist_twist_score)

    right_posture_score_a = find_rula_posture_score_a(upper_arm_score=img_righ
t_upper_arm_score,
                                                      lower_arm_score=img_righ
t_lower_arm_score,
                                                      wrist_posture_score=img_
right_wrist_score,
                                                      wrist_twist_score=img_ri
ght_wrist_twist_score)

    print('Finding posture score for group B')
    print(img_neck_score, img_trunk_score, img_leg_posture_score)

    posture_score_b = find_rula_posture_score_b(neck_posture_score=img_neck_sc
ore,
                                                trunk_posture_score=img_trunk_
score,
                                                leg_posture_score=img_leg_post
ure_score)

    print('Finding grand score for left side')
```

```
    print(left_posture_score_a, posture_score_b)

    left_grand_posture_score = find_rula_grand_score(upper_body_posture_score=
left_posture_score_a,
                                                     lower_body_posture_score=p
osture_score_b)

    print('Finding grand score for right side')
    print(right_posture_score_a, posture_score_b)

    right_grand_posture_score = find_rula_grand_score(upper_body_posture_score
=right_posture_score_a,
                                                       lower_body_posture_score
=posture_score_b)


    rula_final_grand_score = max(left_grand_posture_score, right_grand_posture
_score)

    print('Rula Grand Score ' + str(rula_final_grand_score))

  print('Completed')
test_img_folder_list = [join(test_img_root_path, f) for f in listdir(test_img_
root_path) if not isfile(join(test_img_root_path, f))]
print('No of folders in test image directory: '+str(len(test_img_folder_list))
)
for test_img_folder in test_img_folder_list:
  test_img_path_list = [join(test_img_folder, f) for f in listdir(test_img_fol
der) if isfile(join(test_img_folder, f))]
  for i in range(2):
    img_view_flag = test_img_path_list[i].rsplit(sep='.',maxsplit=1)[0].rsplit
(sep='_',maxsplit=1)[1]
    if img_view_flag=='F':
      front_img = test_img_path_list[i]
    else:
      img_view = img_view_flag
      side_img = test_img_path_list[i]
  start_time_1 = time.time()
  plot_estimate_rula_score(front_img, side_img, img_view)
  elapsed_time_1 = time.time() - start_time_1
  print('Time taken to inference posture: ' + str(elapsed_time_1) + ' seconds'
)
```

Vita Auctoris

NAME:	Gourav Kumar Nayak
PLACE OF BIRTH:	Bhilai, India
YEAR OF BIRTH:	1993
EDUCATION:	BSP Senior Secondary School Sector-10, Bhilai, India, 2007-2011 Chhattisgarh Swami Vivekanand Technical University, B.E., Bhilai, India, 2011-2015 University of Windsor, M.A.Sc., Windsor, ON, 2019-2021