Application of ANN and transfer learning for prevention of Work-related Musculo-Skeletal Disorders (WMSDs) in construction workers

Final Project Report

Srijeet Halder, James Xuan Dung Nguyen, Saeid Alimoradi

Abstract

Work-related Musculo-Skeletal Discorders (WMSD) are defined as a group of painful disorders of muscles, tendons, and nerves. These disorders can cause temporary and/or permanent disabilities and seriously affect workers' livelihood. Previous research has applied Machine Learning (ML) for recognition of WMSD risk. However, previous research used inertial sensors strapped to the body to measure the angle of the body parts. These sensors are expensive and uncomfortable to wear while working. Our project aims to eliminate the need of additional hardware through the use of computer vision. In this project, we trained a ML model to identify WMSD risk from an image of a human. Artificial training data was generated through the use of virtual 3D model of a human in Unity. The human model was posed in different postures mimicking common construction tasks. Small video clips were recorded by rotating the scene camera in Unity around the human model. These clips were then processed with Mediapipe for feature extraction and dataset generation. The generated dataset was then used to train an Artificial Neural Network (ANN) using TensorFlow. A part of the data (20%) was kept for the validation purpose. The choice of hyper-parameters were optimized by implementing the grid search optimization method. After fine-tuning the hyper-parameters, 100% training and validation accuracy was achieved. Finally the trained model was tested on real-life videos of construction workers and found to be performing satisfactorily.

1 Problem statement

Work-related Musculo-Skeletal Disorders (WMSD) is highly prevalent among the construction workers as compared to other industry workers. This is because construction workers typically work in unfinished environments and in odd positions. Pulling, pushing, and carrying physical objects and loads are highly prevalent among construction workers, and account for 37% of WMSDs [1]. The long-term effects of non-ergonomic working postures render many construction workers unfit to continue working in construction [2]. This creates a shortage of experienced workers in the market, which in long run negatively affects the construction industry as well as the economy at large. In 2018, worker's compensation and medical expenditure due to WMSDs was estimated to be $19 billion [3]. Maintaining the health and safety of the workers is not only a business issue but also a moral issue.

Previous research [2] [3] [4] has found that many chronic WMSDs can be prevented by promoting the use of correct postures in construction workers. Previous research has tried to recognize risk of WMSD by analyzing the body postures of the workers. Their approach was based on strapping Inertial Measurement Unit (IMU) sensors to the workers' body. This approach has been considered intrusive and costly [5]. Wearing sensors can also cause discomfort when working [5]. Therefore, this study investigates the use of computer vision for detecting risk of incorrect postures during construction activities and warn the workers of potential WMSD.

2 Methodology

2.1 Dataset

The training dataset for this project was generated through artificial data. For this purpose, the Unity game engine was used. Unity, primarily a game development software, is also a powerful tool for 3D simulation and visualization [6]. It has been used by many researchers for Augmented/Virtual reality research as well [7].

A 3D construction worker model was imported in Unity from the Unity Asset store. The 3D model can be manipulated to stand, sit, or bend in any way as a real human can be. The 3D model was posed in various ways performing common construction tasks, such as painting, sawing, lifting, etc. The human model stayed static while the scene camera of Unity revolved 360°around it. The camera view was recorded which was later used to create the data-set. Figure 1 shows few frames from two of the recorded videos. The videos were labelled as right and wrong postures. Total 5 right postures and 9 wrong postures were recorded from the subjective opinion of the project team.

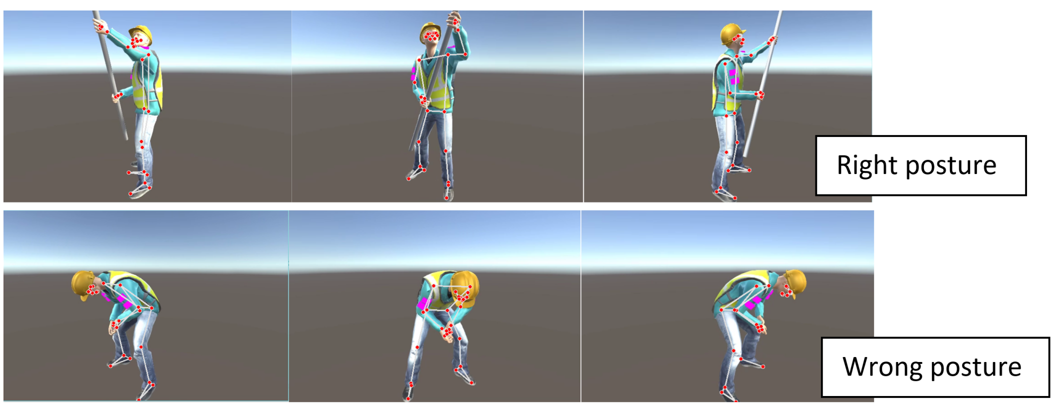

Figure 1: Artificial images used for training

Having exported the videos in .mp4 format, the coordinates representing the joints of the worker's body were extracted using the Mediapipe [8] library developed by Google. The Mediapipe identifies 33 landmarks on a human body. For each video, each frame was extracted using the OpenCV-Python library. Each frame was passed on to the Mediapipe's PoseDetector class. The PoseDetector class returned the coordinates of each of the landmarks. The returned coordinates are normalized from 0 to 1 by Mediapipe itself. Three coordinates (x,y,z) are returned for each landmark. Therefore, for each frame 99 numerical values are generated that ranges from 0 to 1. These landmarks are plotted over the video, which can be seen in Figure 1.

This is an example of transfer learning where a pre-trained model is used to extract features for another model. The advantage of using transfer learning is that the Mediapipe library was trained using more than 100K images in variety of situations and using the knowledge stored in the model allowed us to solve our problem with a very small set of training videos.

Since the postures are either right or wrong, the Machine Learning task is of binary classification type. Accordingly, as explained above each of the videos was categorised into two labels - right and wrong. The program processed each video one by one, extracted the landmark coordinates, and appended them into a 2D list. After processing of all the videos, the list is written into a CSV file, which is later used as the training dataset for the machine learning model. A sample of the final numerical dataset used for training is shown in Figure 2.

```
         0    1    2         3  ...        96        97        98  label
687    0.0  0.0  0.0  0.001391  ...  0.193415  0.578867  0.027509    1.0
7827   0.0  0.0  0.0  0.009344  ... -0.024631  0.325088  0.528286    0.0
9398   0.0  0.0  0.0 -0.001797  ...  0.199932  0.109255  0.717377    0.0
10479  0.0  0.0  0.0 -0.006628  ... -0.011499  0.395163  0.162787    0.0
7531   0.0  0.0  0.0 -0.005625  ...  0.179728  0.273249 -0.310860    0.0

[5 rows x 100 columns]
```

Figure 2: Sample dataset

2.2 Training

The machine learning model used in this project was an Artificial Neural Network (ANN). The popular open-source ANN framework called TensorFlow, initially developed by Google, was used to train the neural net.

The Neural Net was designed with only 1 hidden layer which was later found to be sufficient. The initial configuration had 30 neurons in the hidden layer. A dense layer type was used without any activation function for the hidden layer. The number of input nodes corresponded with the number of features, i.e., 99 (33 landmarks x 3 x,y,z coordinates). There was only 1 output neuron in the output layer. The labels ’right’ and ’wrong’ were converted into numerical form as 1 (right) and 0 (wrong). A ’sigmoid’ activation function was applied to the output layer, which is the generally recommended activation function for binary classification problems in Neural Nets [9]. The training was performed for 40 epochs initially with the default batch size of 3. The Adam optimizer and binary_crossentropy loss function were used for the training. Additionally, the data-set was also split into 80/20 train-test split before training using the train_test_split function from Scikit-learn (a machine learning library for Python). Some snippets of code for data pre-processing and model training are shown in Figure 3 and 4.

```
X = training_dataset.drop('label',axis=1).values
X = np.asarray(X).astype('float32')
Y = training_dataset[['label']].values
Y = np.asarray(Y).astype('float32')

X_train, X_test, Y_train, Y_test = train_test_split( X, Y, train_size=0.8)
```

Figure 3: Code snippet: data pre-processing

```
model = Sequential()
model.add(Dense(30,input_dim=99))                   #hidden layer
model.add(Dense(1,activation='sigmoid'))                   #output layer
model.compile(loss='binary_crossentropy',metrics=['Accuracy','Precision','Recall','FalsePositives','FalseNegatives'])

model.fit(
    X_train,
    Y_train,
    epochs=40,
    verbose=2,
    callbacks=[logger],
    validation_data=(X_test,Y_test)
)
model.save("trained_model.h5")
```

Figure 4: Code snippet: model training

2.3 Validation and Optimization

The training process took about 50.15 seconds in the initial run. Later the authors implemented the grid search optimization method to fine-tune the hyper-parameters. The tuned hyper-parameters and the range of values tested are shown in Table 1.

Table 1: Hyper-parameter tuning range

Parameter	Range
Learning rate	[0.01, 0.05, 0.1]
Hidden neurons	[20,50,70]
Optimizer	[SGD, RMSprop, Adam]
Epochs	[50,80,100,150]
Hidden-layer activation	['relu',None]

With the above test configurations, the best outcome was achieved with learning_rate=0.01, hidden_neurons=20, epochs=50, optimizer=RMSprop, and hidden-layer activation=ReLU. With this configuration, the accuracy achieved was 100% and total training time was reduced to 14.67 seconds. The validation accuracy of 100% was also achieved with the this configuration, which indicates no model over-fitting. The other training results are shown in Table 2.

Table 2: Training results

Evaluation Metric	Value
Loss	8.09e-06
Accuracy	1.000
Precision	1.000
Recall	1.000
Training time	14.67s

3 Testing

After the optimization step, the model was tested with real-life videos of construction workers. The testing approach was more qualitative than quantitative. The main purpose of the testing was to ensure that the model trained with artificial data can be applied to real-life images and videos.

Two videos of construction workers performing flooring and cleaning activities were downloaded from online public sources. The videos were processed frame-by-frame using the OpenCV-Python library and the PoseDetection module of Mediapipe to extract the feature vector. Then, the feature vector was passed to the model.predict() function of TensorFlow for classification.

The model gave the output as 0 to 1, which can be interpreted as the "belongingness" of the input to either of the class. A number close to 0 is more closely related to the class 0, which in this case is the label 'wrong'.

We defined WMSD risk as $(1 - prediction) * 100$. For example, a model prediction value of 0.3 will be interpreted as 70% risk of WMSD, a model prediction of 0.6 is interpreted as 40% risk of WMSD, and so on.

An indicator was also added as shown in Fig. 5 and 6 to visually show the risk of WMSD from the observed image. The model was found able to label bad postures like bending down and couching as wrong, while labelling good postures like standing straight as right. Some example frames from the test videos are shown in Fig. 5 and 6. The model could be deemed as performing satisfactorily. The major achievement of the study was in training a model with no real data that performed reasonably well with real test data.

4 Conclusion

This study is a proof-of-concept study of using computer vision for detecting risk of WMSD in construction workers. This study also showed the potential of using transfer learning through the use of a pre-trained model in the form of Mediapipe library for solving a separate problem without the need of a huge training dataset. The knowledge stored in the Mediapipe models was reused for this purpose which saved effort in creating new dataset in variety of background settings and with diverse

Figure 5: Sample test image 1

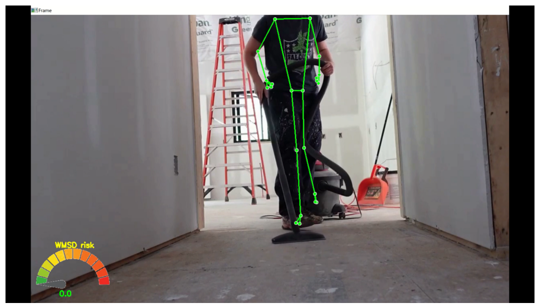

Figure 6: Sample test image 2

human body structure. Also ANN was found to be an useful ML technique that produced very high accuracy with very less training time.

In this study, the authors did not make any attempt on identifying the right or wrong postures. Authors labelled the postures based on their subjective opinion.

Future research may load this model in an autonomous robot that monitors the construction site and give verbal feedback to construction workers to educate them on the WMSD risk.

5 Contributions

Group Member	**Contribution**
Saeid Alimoradi	Data pre-processing, Optimization, Writing
Srijeet Halder	3D human models, ANN Training, Testing, Writing
James Xuan Dung Nguyen	Literature review, Writing, Editing

References

[1] X. Yan, H. Li, A. R. Li, and H. Zhang, "Wearable imu-based real-time motion warning system for construction workers' musculoskeletal disorders prevention," *Automation in construction*, vol. 74, pp. 2–11, 2017.

[2] X. Wang, X. S. Dong, S. D. Choi, and J. Dement, "Work-related musculoskeletal disorders among construction workers in the united states from 1992 to 2014," *Occupational and environmental medicine*, vol. 74, no. 5, pp. 374–380, 2017.

[3] A. A. Akanmu, J. Olayiwola, O. Ogunseiju, and D. McFeeters, "Cyber-physical postural training system for construction workers," *Automation in Construction*, vol. 117, p. 103 272, Sep. 2020, ISSN: 09265805. DOI: `10.1016/j.autcon.2020.103272`.

[4] T. Cheng, G. C. Migliaccio, J. Teizer, and U. C. Gatti, "Data fusion of real-time location sensing and physiological status monitoring for ergonomics analysis of construction workers," *Journal of Computing in Civil engineering*, vol. 27, no. 3, pp. 320–335, 2013.

[5] V. C. Chan, G. B. Ross, A. L. Clouthier, S. L. Fischer, and R. B. Graham, "The role of machine learning in the primary prevention of work-related musculoskeletal disorders: A scoping review," *Applied Ergonomics*, vol. 98, p. 103 574, 2022.

[6] S. Halder, K. Afsari, J. Serdakowski, and S. DeVito, "A Methodology for BIM-enabled Automated Reality Capture in Construction Inspection with Quadruped Robots," in *Proceedings of the 38th International Symposium on Automation and Robotics in Construction (ISARC)*, Dubai, UAE, Nov. 2021, pp. 17–24, ISBN: 978-952-69524-1-3.

[7] H. C. Pham, N. Dao, A. Pedro, *et al.*, "Virtual field trip for mobile construction safety education using 360-degree panoramic virtual reality," *Int. J. Eng. Educ*, vol. 34, no. 4, pp. 1174–1191, 2018.

[8] *MediaPipe*. [Online]. Available: `https://mediapipe.dev/` (visited on 10/26/2021).

[9] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. 2019.