Convolutional Long Short-Term Memory Model for Recognizing Postures from Wearable Sensor

Junqi Zhao, Esther Obonyo
Pennsylvania State University, U.S.
eao4@psu.edu

Abstract. This research investigates the feasibility and viability of applying Deep Neural Networks (DNN) to improve performance with respect to posture recognition based on multi-channel motion data from Wearable Sensors (WS). The authors use the recognition of posture that can be linked to risk of Musculoskeletal Disorder (MSD)- among construction workers as the testbed. The proposed approach is based on the use of a DNN model integrating Convolutional Neural Network (CNN) and Long short-term memory (LSTM) that can achieve automated feature engineering and sequential pattern detection. The model performance was evaluated using datasets collected from four construction workers. The proposed model outperformed baseline CNN and LSTM models. Under the personalized modelling approach, it improved recognition performance by 3% from the benchmark Machine Learning models; the improvement is 2% for generalized modelling approach. The proposed model achieves high-performance posture recognition, which facilitates the MSD prevention in construction through monitoring injury-related postures.

1. Introduction

The authors explore the feasibility and viability of using DNN to improve performance in the recognition of motion data from multi-channel Wearable Sensor (WS) – this data can be treated as high-dimensional data streams (Plötz and Guan 2018). It has been demonstrated that Deep Neural Networks (DNN) models can increase performance in the recognition of objects from high-dimensional data streams. Successful deployments have been in application domains such as Computer Vision, Speech Recognition, and Natural language Processing (Plötz and Guan 2018). Building environment examples include the use of DNN models in the automated objects detection, ranging from pipeline cracks for structure health monitoring (Cheng and Wang 2018) to workers' behaviour monitoring (Xiaochun Luo et al. 2018b). There is an opportunity to deploy DNN models to the detection of awkward postures that compound the risk of workers developing musculoskeletal disorders (MSD) (Zhao and Obonyo 2018). Examples of closely related work research done by Chen et al. (2017) and Ryu et al. (2018). This study focuses specifically on the extent to which DNN-based models can be used to achieve high posture recognition performance using motion data from WS.

An ideal solution is one that can be used to improve the accuracy with respect to the recognition of awkward posture without increasing the computational effort. The use of Machine-Learning (ML)-based models in detecting MSD-related posture from WS output has the advantages of robustness in challenging deployment contexts and ease of data processing (Chen et al. 2017, Ryu et al. 2018, Zhao and Obonyo 2018). It is important to note that conventional ML-based models use the "sliding-window-based analysis pipeline" (Plötz and Guan 2018). The main problems associated with the use of this approach include: i) heuristic engineering biases, ii) ignoring the sequential patterns, and iii) isolation of feature engineering and selection, as well as model optimization. These problems result in the model's performance being sub-optimal. The authors contend that this challenge can be addressed through leveraging DNN's deep hierarchical architecture - more specifically the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN). CNN can extract rich features from raw output without manual feature engineering while the Recurrent Neural Network (RNN) can be used to improve the recognition of temporal patterns (Plötz and Guan 2018). It has been demonstrated that the use of a deep hybrid model that integrates the CNN and RNN can result in improved performance with respect to detecting workers' activities from videos (Ding et al. 2018). In the subsequent sections, this research investigates the feasibility of adapting and replicating this approach to posture recognition that can be linked to the risk of MSD among construction workers. The remainder of the paper is organized as follows. Section 2 briefly reviews the research background. Section 3 describes the proposed DNN model configuration and model test. The test result and discussion are reported in Section 4. The conclusion and further work summarized in Section 5.

2. Theoretical Background

2.1 Posture-based MSD Assessment and Prevention

Epidemiological studies have established that physical factors pose the highest risks for MSD (Nunes and Bush 2012). Workers in labor-intensive sectors such as construction routinely adopt awkward working postures. This exposes them to a high risk of developing MSD. Some of these efforts directed at addressing this problem through the development of more proactive safety management practices have focused on the use of emerging wearable sensor technologies to monitor the posture adopted by the targeted workers. Such efforts are at times leverage ergonomics rules to facilitate the monitoring of repetitive awkward postures. Common ergonomic rules include "Rapid Entire Body Assessment" (Hignett and McAtamney 2000), Ovako Working Posture Analyzing System (OWAS) and its extension (Kivi and Mattila 1991), and the ISO 11226:2000 (Normalización 2000). The use of such rules in construction relies on the visual assessment of superintendents – these tend to suffer from subjective biases.

2.2 Posture Capture in Construction

Vision-based sensing and wearable sensing techniques present an opportunity for motion data collection in a more effective and efficient way, compared to manual inspection (Wang et al. 2015). Vision-based sensing has exceptionally high accuracy and often comes with powerful analytical tools for assessing both body posture and internal joint load (Han and Lee 2013). However, the vulnerability to occlusion and lighting conditions, as well as data processing complexity (Ding et al. 2018, Hanbin Luo et al. 2018) limits its application to construction sites.

Wearable sensors, particularly the Inertial Measurement Units (IMU), are more applicable to the constraints of working in a construction job site. IMU capture streaming motion data in a non-intrusive, robust, and cost-effective manner (Wang et al. 2015). The raw IMU sensor output can be used to detect awkward posture without the need for a total reconstruction of a 3D human body model. Such advantages are brought by the ML-based posture recognition models. The robustness and flexibility of IMU sensors make them appropriate for use on the construction job site.

2.3 DNN for Posture and Activity Recognition

Conventional ML-based models suffer from three main problems when used in pattern recognition tasks (Plötz and Guan 2018). Firstly, heuristic feature engineering, which manually crafts features, is a time-consuming and biased feature construction process. Secondly, it separates sliding windows, isolates the consecutive sample data and assumes that the time-series motion data are static, which neglects any existing sequential patterns. Thirdly, it separates feature engineering and selection from model tuning during the training process, resulting in sub-optimal performance.

DNN models can address some of these challenges. They have been used successfully in pattern recognition tasks in Computer Vision and Speech Recognition (Plötz and Guan 2018). The multi-layer CNN can automatically extract rich features with increasing complexity from raw data, reducing the bias in manual feature engineering. The RNN has the advantage in capturing sequential patterns. The integration of the CNN and RNN leverages the learning capabilities from both extremely complex features and sequential patterns directly from input data. This ultimately improves the performance of the models.

The use of Computer Vision and DNN for posture recognition in construction research is an emerging area. Zhang et al. (2018) used multilayer CNN to extract view-invariant features from a single video camera for awkward working posture recognition, which significantly improved the model's generality. Hanbin Luo et al. (2018) eliminated the need for reconstructing complex 3D human body models in their posture recognition system, that was developed using a pre-trained CNN model (VGG-16). Ding et al. (2018) deployed the RNN with pre-trained CNN model (Inception V3) to further capture the sequential patterns from the video frames. The resulting hybrid model outperformed the state-of-the-art heuristic feature engineering approaches with respect to activity recognition. In other efforts, it was demonstrated that DNN models can be used detect the misuse of Personal Protective Equipment from multiple workers in the same video scene (Fang et al. 2018a, Fang et al. 2018b). The CNN-based classifier (ResNet-50) can also be used to detect construction equipment, materials, and workers, and recognize interactive construction activities (Xiaochun Luo et al. 2018a). Xiaochun Luo et al. (2018b) demonstrated the feasibility of using the CNN model to detect objects from surveillance cameras. These breakthroughs have been made possible because of two key factors: i) the existence of a robust CNN architecture (such as VGG-16, Inception, and ResNet-50) for automated feature extraction from images and videos; ii) RNN for capturing sequential patterns. Notwithstanding these developments, environmental constraints and the insufficient data for training deep model continue to limit the real-life deployment and scaling of DNN models that use data from vision-based sensing (Ding et al. 2018, Hanbin Luo et al. 2018).

This paper explores the possible strategies for addressing these challenges based on the use of data captured using Wearable IMU sensors. The proposed approach treats motion data from multiple sensor channels that has been segmented into windows of the same dimension as 2D "images." This "trick" enables one to apply DNN models to WS output. A simple DNN model was first introduced into wearable sensing-based posture recognition in an effort to find discriminative features (Plötz and Guan 2018). On-going research work has achieved promising recognition performance (Ordóñez and Roggen 2016, Zeng et al. 2014). The DNN-based models have shown high-performance in recognizing human daily activities and postures from WS output. However, there is no universal, pretrained, and ready-to-use deep model architecture for different scenarios (Plötz and Guan 2018). It is, therefore, necessary to investigate how one can configure a DNN-based model for use in the recognition of construction workers' posture.

3. DNN-based Model Development and Test

The DNN models were implemented using Keras (TensorFlow GPU backend). The model development, training, and testing were done on a Windows 10 PC (Intel Core i7-7700 CPU@ 2.8 GHz, 16GB RAM, NIVIDA GeForce GTX 1060 GPU@16GB RAM). The authors experimented with the use of a DNN-based posture recognition model, which combines CNN and Long short-term memory (LSTM) into a Convolutional-LSTM Network (CLN). The LSTM extends the conventional RNN's ability to capture longer temporary patterns. The CLN was evaluated using both non-recurrent CNN and non-convolutional LSTM as baseline models. The difference between CLN and CNN is the topology of fully connected dense layers, the CLN uses LSTM units as dense layers. The LSTM model uses raw sensor output without features learned from CNN. In this case, the difference in model performance can be attributed to the architecture of the model – the performance is not driven by optimization, pre-processing or ad hoc customization (Ordóñez and Roggen 2016).

3.1 Components in DNN-based Model Architecture

A CLN model example using one-layer CNN and one-layer LSTM is shown in Figure 1.

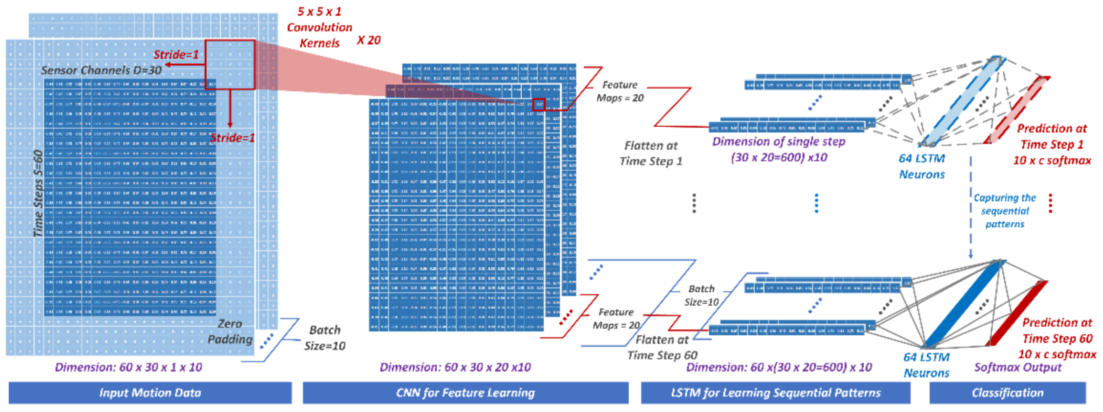


Figure 1 Simple CLN Model Example

Model Input: Model input is segmented by a sliding window. The raw sensor output is normalized for each channel. After segmentation, it can be treated as a 2D image of "S" by "D" (e.g. 60 by 30 in Figure 1), where S is timesteps, and D is the number of sensor channels in a window. All sensor channels are treated as one layer, resulting in an input depth of 1-layer.

Batch and Epoch: The entire dataset for the DNN is divided into multiple (non-overlapping) groups of equal size. One can then feed each group into the model for training. Each group is also referred to as a **batch** (e.g. 10 in Figure 1) for effective model training. One **epoch** of training means all batches of training data pass both forward and backward through the model for once. Multiple epochs can be used for parameter optimization when training data is limited.

Convolutional Layer: A convolutional layer computes the output that is connected to the local region of each sample in the input. The **stride** (e.g. 1 by 1 in Figure 1) quantify the movements of the convolutional kernel along with the vertical or horizontal direction. **Zero-padding** of the input data avoids losing information on the border of 2D input. The reference to "n" convolutional kernels, identifies the number of feature maps (e.g. 20 in Figure 1). This creates an additional dimension of **depth** for the convolutional layers. A **Flatten Layer** is used to establish a full-connected dense layer. It converts each sample's feature maps into a one-dimensional vector representing one sample to be classified.

LSTM Layers: Flattening the CNN output ignores temporal dependencies between different time steps. LSTM can address this problem. Figure 1 shows only the feature maps along the depth dimension are flattened. The vertical time step dimension is reserved for capturing the sequential pattern. Each slice over the time step has a dimension of 600 (features) × 10 (batch size). Samples in a batch fully connect with 64 neurons in LSTM layer. The LSTM neurons are then fully connected with the ***softmax*** layer. This is used to predict the class of each sample in a batch. The LSTM gives a prediction for every time step "t" in sequence. However, the memory of LSTM units tends to become more informed with more time steps pass. This is because the activation information in LSTM neurons at each time step is passed on to the next. This implies that the more time steps LSTM neurons have "seen", the more informative the model can be (Ordóñez and Roggen 2016). Class probability distribution at the last time step "T" is used as a recognition result, when the full sequence in the window have been observed.

Fully-Connected Layer: In the non-recurrent CNN model, all elements in the vector generated by Flatten Layer will connect with neurons in a subsequent fully-connected layer. The last fully-connected layer has an equal number of neurons to the number of class labels. The ***softmax*** function is used to deliver a class probability distribution of samples in the batch. Each sample can be classified by the class label with the highest probability.

3.2 Model Architecture

CLN Model: The proposed CLN model uses four convolutional layers to extract complex features. Each layer has 64 kernels with a size of 5 by the number of sensor channels, 1×1 stride, and zero-padding. Karpathy et al. (2015) and Ordóñez and Roggen (2016) recommend the use of a two-layer LSTM. Each LSTM layer has 128 units. The output from the LSTM is used by the softmax layer for prediction. This model can be expressed as $\boldsymbol{C(64) - C(64) - C(64) - C(64) - RL(128) - RL(128) - Sm}$ (Pigou et al. 2018), where the $\boldsymbol{C(n_c)}$ denotes a convolutional layer with n_c features, $\boldsymbol{RL(n_r)}$ denotes a recurrent LSTM layer with n_r units, $\boldsymbol{Sm}$ is the softmax classification. The hyperbolic tangent function was used in the proposed model to activate neurons in each layer of CNN and LSTM.

Baseline Models: The baseline CNN model was developed by substituting the LSTM layers with two 128-unit dense layers in CLN. The CNN model can be expressed as $\boldsymbol{C(64) - C(64) - C(64) - C(64) - D(128) - D(128) - Sm}$, where $\boldsymbol{D(n_d)}$ denotes a dense layer with n_d units. The baseline LSTM model can be represented as $\boldsymbol{RL(128) - RL(128) - Sm}$. It uses normalized sensor output in each window as model input – it does not use feature engineering.

3.3 Data Collection and Preparation

Four subjects were recruited from nearby construction projects. Five IMU sensors (Mbinet Lab Meta Motion C) were deployed at the hardhat, upper arm, chest center, right thigh, and right calf (Figure 2) by sticking on cloth. After acquiring the workers' consent using Institutional Review Board (IRB)- approved protocols, each subject was asked to perform their routine tasks for 30 minutes. Their activities were recorded for cross-referencing. The data collected are summarized in Table 1.

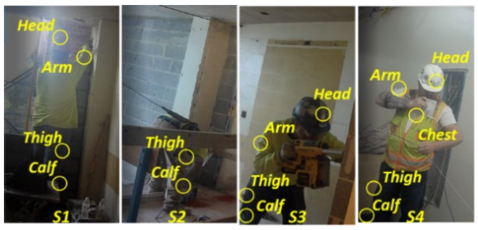


Figure 2. Subjects Working with Sensors (the sensors blocked are not circled)

The output was down-sampled to 40 Hz from all the sensors' channels for S2, S3, and S4. The data for S1 was collected at 25 Hz and down-sampled to 20 Hz. Each record was labelled with video reference. This research used a 1-second window with 50% overlap for segmentation. Each window was labelled using the majority label in the window.

Table 1 Description of Field Dataset (Detailed description provided in Appendix 1)

Subjects	Activities Labels with Proportion	Label Explanation
S1 Masonry	BT (14.7%), KN (2.0%), LB (12.3%), OW (7.2%), ST (52.4%), WK (3.4%)	BT-Static bending and minor movement with bending, and short-term pick up. KN-Kneel on one leg and both legs OW-Overhead Work with one arm or both arms SQ- Squatting, ST-Standing, LB-Literal Bend. NULL-transit movements. The posture data without video reference (due to block) was not labelled, resulting in labelled postures not adding up to 100%
S2 Demolition	BT (72.9%), NULL (4.3%), ST (12.5%), WK (9.1%)	
S3 Electrician	BT (13.6%), KN (46.7%), NULL(15.0%), SQ (3.0%), ST (22.0%)	
S4 Electrician	BT (12.3%), ST (71.5%), WK (12.3%)	

3.4 Set-up for the Model Training and the Model Implementation

Dataset Splitting for Model Training and Testing: Stratified random shuffle was used to split the dataset. This ensured that the same class distribution was maintained in both train and test data. 8:2 ratio is the recommended "rule of thumb" for splitting the datasets in related studies (Ordóñez and Roggen 2016). This set-up was used to fully train the DNN models. The ratio used for the training and validation was set at 9:1 (higher than the recommended ratio). The splitting was repeated five times using different random state. This minimized bias in dataset splitting and improved the reliability of model evaluation.

Model Performance Evaluation Metric: The construction workers' postures are highly unbalanced between classes. This reflects the natural human postures. The classification accuracy is insufficient for measuring model performance - a naive model would achieve high accuracy by classifying every sample as the majority class. The authors used the Macro F1-score to account for this imbalance. The F1-score for each label was acquired using the harmonic mean of Precision and Recall. By giving equal weights to the majority and minority labels when evaluating overall performance for all labels, the model can be trained to achieve high-performance for recognizing all postures. The Macro F1-score is calculated in Equation (1) using an unweighted average. "N" in the equation denotes the number of class labels. High Macro F1- scores reflect high classification performance.

$$\text{Macro F1} = \frac{1}{N}\sum_i 2\frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \qquad \text{Equation (1)}$$

Checkpoint in Model Training: The model performance does increase consistently after every epoch during the training process. The models were trained until their performance ceased to improve. This occurred after *300 epochs*. The model training checkpoint was set to save the trained model with improved performance in an "overwritten" way. It saved the DNN models with the best performance after all epochs. A dropout operation (50%) was used before fully-connected layers to control model overfitting.

4. Result and Discussion

4.1 Model Performance Evaluation

The evaluation was based on the use of conventional ML-based models as the benchmark on the same dataset. The features used in the benchmark models were the same as the authors' previous experiments (Zhao and Obonyo 2018). Classification algorithms in ML-based models include Support Vector Machine (SVM), K-Nearest-Neighbour (KNN), Naive Bayes (NB), Decision Tree (DT), and Random Forest (RF) as an ensemble model. The differences between DNN and ML models are features constructed and classification models used. The ML models were implemented using Scikit-learn (Pedregosa et al. 2011). Results are given in Table 2.

Table 2 Performance Evaluation of Deep Architectures (Macro F1 Score as a metric)[1]

Models	**S1**	**S2**	**S3**	**S4**
CLN	**0.872**	**0.784**	**0.898**	0.846
CNN	0.872	0.783	0.891	0.771
LSTM	0.826	0.772	0.880	**0.855**
Best Benchmark	0.869	0.664	0.840	0.823

The recognition model's performance for each subject reflects the "personalization" capabilities of the proposed approach, which can be used to capture individual posture idiosyncrasy (Plötz and Guan 2018). Table 2 shows the proposed CLN model consistently outperformed the benchmarking models by an average of 3%. However, CNN and LSTM models were outperformed by the benchmarking model for S4 and S1. This supports the hypothesis that the CLN model can leverage the advantages from both the CNN and LSTM layers. The result is consistent with previous work which integrated CNN and RNN models in vision-based (Ding et al. 2018) and WS-based (Ordóñez and Roggen 2016) studies.

Confusion matrixes are constructed to assess model errors (Appendix 2). LSTM models make most classification errors when distinguishing postures between bent, working overhead, and standing (S1). The models also make errors in recognizing transitional postures (S2 and S3). This may be due to the lack of convolutional layers, which limits the LSTM models in learning complex features. The sequential patterns alone are not enough for effective postures recognition. The authors contend that proper feature engineering can be used to improve the models' performance. The proposed CNN model's low performance for S4 stems from the high recognition errors associated with walking posture. This may be caused by data imbalance. S4 contains least postures types and an imbalanced postures distribution. The imbalanced dataset may lead to the overfitting for majority postures and low performance for recognizing minority.

Improved recognition performance contributes to the reliability of posture-based MSD risk assessment. The frequency of awkward postures can be detected more accurately, which enhances the validity of MSD risk level assessment using OWAS. The recognition accuracy greatly influences the ease with which postures can be detected continuously (Ordóñez and Roggen 2016). Improved accuracy can help avoid problems such as false alarm and missing real-time posture assessment. In subsequent research efforts, the authors will investigate these potential improvements further.

4.2 Analysis of Model Generalization

The use of the deep learning model makes it possible for one to generalize deployment of the author's proposed posture recognition approach to new individuals (Plötz and Guan 2018). Datasets S2 and S4 were downsampled to 20 Hz and combined with S1 as S5[2]. The generalized models used were based on the architecture described in Section 3.2 (see the 4 convolutional

[1] The detailed evaluation result of models is provided in Appendix 3 and Appendix 4.

[2] S2 was not used as it did not contain sensor output from arm.

layers in Figure 4). As shown in Table 3, the CLN model consistently outperformed the baseline DNN models. The best benchmarking model's performance also improved by over 2%. This supports the hypothesis that CLN model can successfully extract common subject-invariant features from different subjects. The observations also indicate that the performance of the model was better than what is observed when the heuristic features engineering approach is used. The baseline performance for DNN models was higher than that for the generalized approach (compare Table 2 and Table 3). This can be attributed to increased training data size.

Table 3 Performance Evaluation of Generalized Model (Macro F1 Score)

	CLN	**CNN**	**LSTM**	**Benchmark (SVM[3])**
S5	0.852	0.813	0.827	0.835

The generalized models' confusion matrices (Figure 3) depicts how the CLN model improves the benchmarking model (SVM). CLN and SVM give a comparable performance in recognizing common postures from multiple subjects (BT, KN, ST, WK). The CLN model exhibited a more superior performance when the recognized postures were limited to only one subject (LB, NON, OW, SQ). The CLN outperformed the benchmarking model by 28% when detecting transitional postures (NULL). This may be explained by the CLN's ability to capture sequential data patterns and identify dynamic postures. The results also show that the CLN model can be used to effectively detect an individual's unique postures even where there is less data.

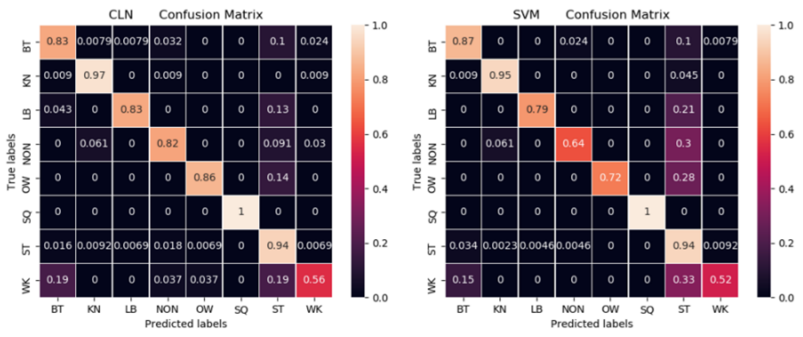


Figure 3 Confusion Matrices for CLN and Benchmarking SVM

4.3 Analysis of Fusing Multi-Sensor Channels

The output from different sensor channels was fused directly based on an assumption that they were similar to pixels in images. The validity of this assumption was assessed using the CLN model's performance when fusing different sensor channels on the S5 dataset. It was established that modifying the convolutional kernel sizes can allow the direct fusion. The performance increased by 1.9% when fusing two types of channels. These findings suggest fusing motion data across channels is an appropriate strategy for proposed CLN model. Accelerometers contribute more to increased recognition performance. Table 4 shows that the model using data from the accelerometer resulted in superior performance than that based on data from the gyroscope. These findings are consistent with observations form Ordóñez and Roggen (2016).

Table 4 Performance Evaluation of Fusing Multi-Sensor Channels (Macro F1 Score)

	Accelerometers	**Gyroscopes**	**Accelerometers+ Gyroscopes**
Convolutional Kernel	20Hz*15channels	20Hz*15channels	20Hz*30channels
CLN Performance	0.836	0.592	0.852

[3] The benchmarking model is SVM with 90 features selected using Recursive Feature Elimination.

4.4 Analysis of Hyperparameter

Convolutional layers' depth influences DNN models' ability to learn complex features. The authors compared the performance of baseline CNN model with varying depths using dataset S5. The goal was to evaluate influences from convolutional layers without considering the influence of the LSTM layers.

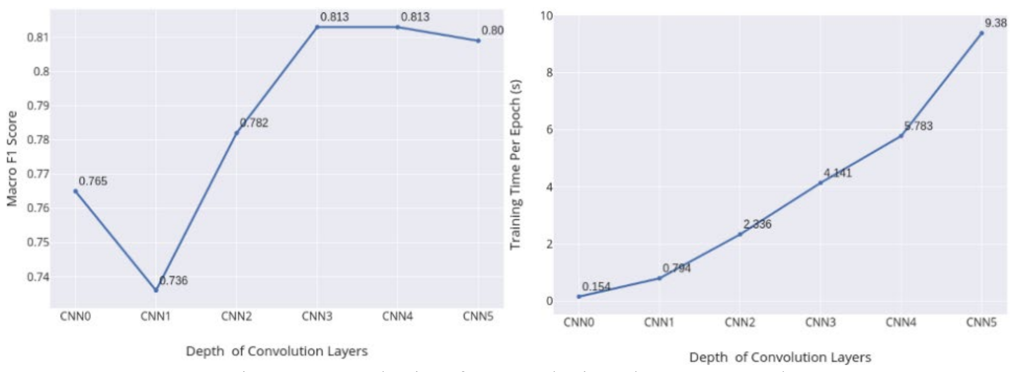


Figure 4 Analysis of Convolutional Layer Depth

As shown in Figure 4, increasing the depth of the convolutional layer from one to three significantly improves the model's performance. There is a plateau when convolutional layer depth reaches four, after which the model's performance starts to decrease. Without the convolutional layers, the tested CNN model becomes a Multilayer Perceptron (MLP) model ($\boldsymbol{D}(\mathbf{128}) - \boldsymbol{D}(\mathbf{128}) - \boldsymbol{Sm}$). The CNN model's performance is no better than the MLP model when only one convolutional layer is used. These results suggest that the convolutional layers can be used to extract features effectively as long as the optimal depth has been identified. The improper feature representation from one shallow convolutional layer can have an adverse impact on the model's performance. As show in Figure 4, too deep of an architecture can decrease the model's performance. Besides the problem of overfitting, the deep architecture can also result in a "varnishing gradient" problem, that is, when the model's performance decreases because of the architecture becoming too complex. Additionally, the model training time increases significantly when it goes deeper. It is necessary to tune the depth for balancing complexity and performance.

5. Conclusion and Future Work

The proposed use of a CLN model can improve the performance of recognizing construction workers' postures based on the use of data obtained from wearable IMU sensors. The consistently high-performance of the CLN model supports that the author's position that: i) the integrated CLN model can take advantage both CNN and LSTM models' power to further improve the DNN-based model performance; ii) an automated feature extraction approach can reduce engineering bias in the heuristic feature engineering process, thus resulting in improved recognition performance.

The CLN model can capture both the "subject-invariant" common features and unique features of a specific subject. This outcome cannot not be easily achieved when using the heuristic feature engineering approach. The CLN model can be a promising approach to balance generalization and personalization (Plötz and Guan 2018). This is an important goal in ML-based posture recognition. The CLN model can also be applied directly to output from WS. The deployed CLN model can learn complex features directly from the raw sensor output. CLN model performance tends to increase with more sensor channels. However, it is recommended to tune the convolutional layer depth as a hyperparameter for DNN-based models. The learning power may not be fully leveraged in shallow and overly deep models.

The authors have extended the application of DNN-based models beyond vision-sensing to multi-channel motion sensing data. The proposed approach was validated using the need for accurate posture recognition in MSD risk assessment based on the use of data derived from WS. In subsequent efforts, the authors will refine the DNN models' hyperparameters and deploy ensembled DNN models for performance improvement. The developed model will be reconfigured for deployment on mobile devices in real-time. The authors will also investigate the CLN performance on "unseen" subject's data.

References

Chen, J., Qiu, J. and Ahn, C. (2017) Construction worker's awkward posture recognition through supervised motion tensor decomposition. Automation in Construction, 77, pp. 67-81.

Cheng, J. C. and Wang, M. (2018) Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. Automation in Construction, 95, pp. 155-171.

Ding, L., Fang, W., Luo, H., Love, P. E., Zhong, B. and Ouyang, X. (2018) A deep hybrid learning model to detect unsafe behavior: integrating convolution neural networks and long short-term memory. Automation in Construction, 86, pp. 118-124.

Fang, Q., Li, H., Luo, X., Ding, L., Luo, H. and Li, C. (2018a) Computer vision aided inspection on falling prevention measures for steeplejacks in an aerial environment. Automation in Construction, 93, pp. 148-164.

Fang, Q., Li, H., Luo, X., Ding, L., Luo, H., Rose, T. M. and An, W. (2018b) Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. Automation in Construction, 85, pp. 1-9.

Han, S. and Lee, S. (2013) A vision-based motion capture and recognition framework for behavior-based safety management. Automation in Construction, 35, pp. 131-141.

Hignett, S. and McAtamney, L. (2000) Rapid entire body assessment (REBA). Applied Ergonomics, 31(2), pp. 201-205.

Karpathy, A., Johnson, J. and Fei-Fei, L. (2015) Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078.

Kivi, P. and Mattila, M. (1991) Analysis and improvement of work postures in the building industry: application of the computerised OWAS method. Applied Ergonomics, 22(1), pp. 43-48.

Luo, H., Xiong, C., Fang, W., Love, P. E., Zhang, B. and Ouyang, X. (2018) Convolutional neural networks: Computer vision-based workforce activity assessment in construction. Automation in Construction, 94, pp. 282-289.

Luo, X., Li, H., Cao, D., Dai, F., Seo, J. and Lee, S. (2018a) Recognizing Diverse Construction Activities in Site Images via Relevance Networks of Construction-Related Objects Detected by Convolutional Neural Networks. Journal of Computing in Civil Engineering, 32(3).

Luo, X., Li, H., Cao, D., Yu, Y., Yang, X. and Huang, T. (2018b) Towards efficient and objective work sampling: Recognizing workers' activities in site surveillance videos with two-stream convolutional networks. Automation in Construction, 94, pp. 360-370.

Nunes, I. L. and Bush, P. M. (2012) Work-related musculoskeletal disorders assessment and prevention. in Ergonomics-A Systems Approach: InTech.

Ordóñez, F. J. and Roggen, D. (2016) Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1), pp. 115.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R. and Dubourg, V. (2011) Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp. 2825-2830.

Pigou, L., Van Den Oord, A., Dieleman, S., Van Herreweghe, M. and Dambre, J. (2018) Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. International Journal of Computer Vision, 126(2-4), pp. 430-439.

Plötz, T. and Guan, Y. (2018) Deep Learning for Human Activity Recognition in Mobile Computing. Computer, 51(5), pp. 50-59.

Ryu, J., Seo, J., Jebelli, H. and Lee, S. (2018) Automated Action Recognition Using an Accelerometer-Embedded Wristband-Type Activity Tracker. Journal of construction engineering and management, 145(1), pp. 04018114.

Wang, D., Dai, F. and Ning, X. (2015) Risk Assessment of Work-Related Musculoskeletal Disorders in Construction: State-of-the-Art Review. Journal of construction engineering and management, 141(6), pp. 04015008.

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P. and Zhang, J. (2014) Convolutional neural networks for human activity recognition using mobile sensors. in 6th International Conference on Mobile Computing, Applications and Services: IEEE. pp. 197-205.

Zhang, H., Yan, X. and Li, H. (2018) Ergonomic posture recognition using 3D view-invariant features from single ordinary camera. Automation in Construction, 94, pp. 1-10.

Zhao, J. and Obonyo, E. (2018) Towards a Data-Driven Approach to Injury Prevention in Construction. in Advanced Computing Strategies for Engineering. EG-ICE 2018,Cham: Springer International Publishing. pp. 385-411.

Appendix

Appendix 1 Description of Collect Dataset

Subjects	Tasks	Collected Data	Used Data	Dataset Size
S1 Masonry	Bricklaying	30.75min @25Hz @30 channels	30.27min @20Hz@30 channels	33260 rows by 30 columns .CSV file 6.54 MB
S2 Demolition	Ground Guardrail Installation	30.64min @50Hz@30 channels	30.27min @40Hz@24 channels[4]	65200 rows by 24 columns .CSV file 10.0 MB
S3 Electrician	Ground Electrical Conduit Installation	26.94min @50Hz@30 channels	18.50min @40Hz@30 channels	44080 rows by 30 columns .CSV file 8.51 MB
S4 Electrician	Wire Pulling	28.48min @50Hz@30 channels	18.50min @40Hz@30 channels	51960 rows by 30 columns .CSV file 10.0 MB

Appendix 2 Performance Evaluation of Deep Architectures

		Model Setup			Model Setup
		Ave_F1	Epoch_time (s)	Exe_time (s)	
S1	CLN	0.872	3.096	1.137	20Hz*30Channels 333 examples in test dataset
	CNN	0.872	2.400	0.730	
	LSTM	0.826	0.682	1.339	
	Best Benchmark	0.869			
S2	CLN	0.784	3.821	1.385	40Hz*24Channels 326 examples in test dataset
	CNN	0.783	2.563	1.056	
	LSTM	0.772	1.272	1.974	
	Best Benchmark	0.664			
S3	CLN	0.898	3.006	2.750	40Hz*30Channels 220 examples in test dataset
	CNN	0.891	2.097	0.324	
	LSTM	0.880	1.070	1.185	
	Best Benchmark	0.840			
S4	CLN	0.846	3.600	2.251	40Hz*30Channels 260 examples in test dataset
	CNN	0.771	2.356	0.330	
	LSTM	0.855	1.997	2.894	
	Best Benchmark	0.823			

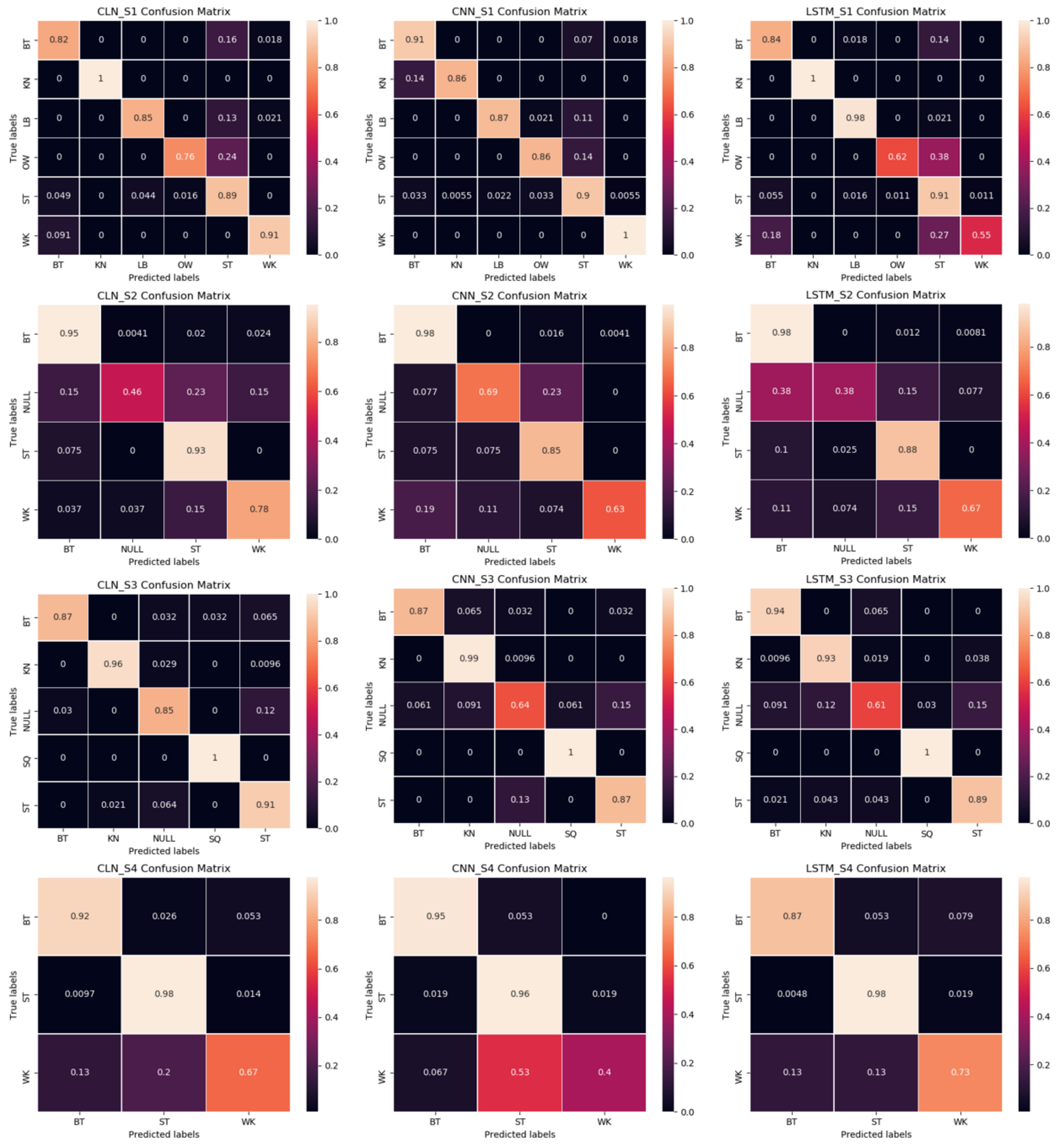
[4] The arm sensor came across malfunction during data collection, the six channels from arm sensor was not considered.

Appendix 3 Performance Evaluation of Benchmarking ML-based Models. FF represent full features, SF represents features selected through Recursive Feature Elimination

		Key Model Parameters Setup						
		Ave_F1		**Train Time (s)**		**Test Time (s)**		
		FF	**SF**	**FF**	**SF**	**FF**	**SF**	
SVM	S1	0.858	0.811	2.912	0.077	0.278	0.007	Kernel: Radial Basis Function Kernel Coefficient: 1/features Multi-Class: One-vs-one Penalty Parameter C: 1
	S2	0.633	0.606	1.307	0.338	0.098	0.026	
	S3	0.826	0.786	1.924	0.033	0.181	0.003	
	S4	0.820	0.809	1.286	0.103	0.088	0.009	
KNN	S1	0.836	0.807	0.053	0.003	0.702	0.014	K: 7[5] Distance Metric: Euclidean Weight: uniform weight
	S2	0.664	0.582	0.038	0.011	0.485	0.100	
	S3	0.785	0.779	0.038	0.005	0.362	0.007	
	S4	0.830	0.799	0.054	0.010	0.697	0.048	
NB	S1	0.568	0.680	0.026	0.002	0.005	0.000	Prior probabilities: classes proportion
	S2	0.622	0.592	0.036	0.008	0.004	0.001	
	S3	0.720	0.758	0.021	0.004	0.002	0.000	
	S4	0.726	0.773	0.050	0.004	0.009	0.010	
DT	S1	0.776	0.707	2.042	0.060	0.001	0.000	Criterion: Gini index Min Samples for Split: 2 Max Depth: No limits Max Leaf Nodes: No limits
	S2	0.643	0.549	0.813	0.360	0.001	0.000	
	S3	0.779	0.667	0.713	0.023	0.001	0.000	
	S4	0.760	0.778	0.920	0.085	0.001	0.000	
RF	S1	0.869	0.782	4.181	0.513	0.011	0.007	Ensemble Method: Bootstrap Number of Estimators: 100 Estimator: DT Estimator Setup: Same as DT
	S2	0.648	0.600	2.564	1.107	0.009	0.007	
	S3	0.840	0.794	1.859	0.262	0.008	0.006	
	S4	0.822	0.823	1.918	0.483	0.008	0.006	

[5] Parameter tuning from cross-validation.

Appendix 4 Confusion Matrix for DNN Models on Each Subject. For each subject, all the three DNN models used are those giving median performance among the five trained models with stratified shuffle splitting


CLN_S1 Confusion Matrix
BT 0.82 0 0 0 0.16 0.018
KN 0 1 0 0 0 0
LB 0 0 0.85 0 0.13 0.021
OW 0 0 0 0.76 0.24 0
ST 0.049 0 0.044 0.016 0.89 0
WK 0.091 0 0 0 0 0.91
BT KN LB OW ST WK
CNN_S1 Confusion Matrix
BT 0.91 0 0 0 0.07 0.018
KN 0.14 0.86 0 0 0 0
LB 0 0 0.87 0.021 0.11 0
OW 0 0 0 0.86 0.14 0
ST 0.033 0.0055 0.022 0.033 0.9 0.0055
WK 0 0 0 0 0 1
LSTM_S1 Confusion Matrix
BT 0.84 0 0.018 0 0.14 0
KN 0 1 0 0 0 0
LB 0 0 0.98 0 0.021 0
OW 0 0 0 0.62 0.38 0
ST 0.055 0 0.016 0.011 0.91 0.011
WK 0.18 0 0 0 0.27 0.55
CLN_S2 Confusion Matrix
BT 0.95 0.0041 0.02 0.024
NULL 0.15 0.46 0.23 0.15
ST 0.075 0 0.93 0
WK 0.037 0.037 0.15 0.78
BT NULL ST WK
CNN_S2 Confusion Matrix
BT 0.98 0 0.016 0.0041
NULL 0.077 0.69 0.23 0
ST 0.075 0.075 0.85 0
WK 0.19 0.11 0.074 0.63
LSTM_S2 Confusion Matrix
BT 0.98 0 0.012 0.0081
NULL 0.38 0.38 0.15 0.077
ST 0.1 0.025 0.88 0
WK 0.11 0.074 0.15 0.67
CLN_S3 Confusion Matrix
BT 0.87 0 0.032 0.032 0.065
KN 0 0.96 0.029 0 0.0096
NULL 0.03 0 0.85 0 0.12
SQ 0 0 0 1 0
ST 0 0.021 0.064 0 0.91
BT KN NULL SQ ST
CNN_S3 Confusion Matrix
BT 0.87 0.065 0.032 0 0.032
KN 0 0.99 0.0096 0 0
NULL 0.061 0.091 0.64 0.061 0.15
SQ 0 0 0 1 0
ST 0 0 0.13 0 0.87
LSTM_S3 Confusion Matrix
BT 0.94 0 0.065 0 0
KN 0.0096 0.93 0.019 0 0.038
NULL 0.091 0.12 0.61 0.03 0.15
SQ 0 0 0 1 0
ST 0.021 0.043 0.043 0 0.89
CLN_S4 Confusion Matrix
BT 0.92 0.026 0.053
ST 0.0097 0.98 0.014
WK 0.13 0.2 0.67
BT ST WK
CNN_S4 Confusion Matrix
BT 0.95 0.053 0
ST 0.019 0.96 0.019
WK 0.067 0.53 0.4
LSTM_S4 Confusion Matrix
BT 0.87 0.053 0.079
ST 0.0048 0.98 0.019
WK 0.13 0.13 0.73
True labels
Predicted labels