University of Cincinnati

Date: 10/29/2021

I, Brennan Thomas, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Science.

It is entitled:

LSTM Neural Networks for Detection and Assessment of Back Pain Risk in Manual Lifting

Student's name: **Brennan Thomas**

This work and its defense approved by:

Committee chair: Rashmi Jha, Ph.D.

Committee member: Fred Annexstein, Ph.D.

Committee member: Ming-Lun Lu, Ph.D.

41563

LSTM Neural Networks for Detection and Assessment of Back Pain Risk in Manual Lifting

A thesis submitted to the

Graduate School

of the University of Cincinnati

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Department of Electrical Engineering and Computer Science

of the College of Engineering and Applied Science

by

Brennan Thomas

B.S. University of Cincinnati

October 2021

Committee Chair: Rashmi Jha, Ph.D

Committee Members: Ming-Lun Lu, Ph.D; Fred Annexstein, Ph.D

Abstract

Repetitive occupational lifting of objects has been shown to create an increased risk for incidence of back pain. Ergonomic workstations that promote proper lifting technique can reduce risk, but it is difficult to assess such workstations without constant risk monitoring. Inertial measurement units (IMU) such as accelerometers and gyroscopes have been used with success in human activity recognition (HAR) systems to determine when specified actions occur, but largely only for general activities for which it is easy to collect a significant amount of data. There has been considerably less work towards assessment of specialized tasks, such as lifting action. This work presents a dual system utilizing machine learning for *detection* of lifting action and *assessment* of the risks involved for that action. The proposed system achieves good performance in both the detection and assessment tasks using raw time-series IMU data with minimal preprocessing. Application of data augmentation provides additional increases in performance for the assessment task, and saliency mapping determines optimal sensor configurations for system modifications. The presented system can be used to monitor the risks involved with lifting action required in a workplace, guiding efforts to mitigate long-term risk.

Keywords: activity recognition, back pain, accelerometer, lstm network, residual network, deep learning, data augmentation, saliency mapping

Acknowledgements

First, I would like to thank Dr. Jha for providing me with guidance and support over the course of my research. I would not have been able to complete this research without her continued suggestions and encouragement.

I would also like to thank Dr. Lu and others at NIOSH for the collection of the dataset used in this thesis, and his continued guidance on this project. His perspective from the ergonomics side helped provide me with a clear goal to strive for.

I would like to thank Kristian Snyder for beginning this work, and providing me with a basis on which to work and become acquainted with research processes.

I would like to thank my parents for always believing in me, and for providing me the support I needed to excel throughout my academic career.

I would like to thank my partner and my friends for their encouragement and for helping me relax during my most stressful moments.

Lastly, I would like to thank my cats Vanilla and Cocoa, for being excellent models and only occasionally stepping on my keyboard while I try to type.

Table of Contents

List of Tables and Figures

Chapter 1. Introduction

1.1 Background

1.1.1 Assessing Lift Risk

Physically demanding jobs requiring strenuous activities such as pushing, pulling, and lifting present serious risk of musculoskeletal disorders to workers, resulting in more forced time off and lost capital than any other occupational injury [1, 2, 3]. These injuries can often lead to permanent disability and are exacerbated by repeated lifting of heavy objects [4]. In 1981, the National Institute for Occupational Safety and Health (NIOSH) introduced the NIOSH Lifting Equation, later revised in 1991 [1, 5]. The Revised NIOSH Lifting Equation (RNLE) provides a limit on the amount of weight that can be safely lifted in a given situation:

$$RWL = LC * HM * VM * DM * AM * FM * CM$$

Where:

- RWL is the maximum weight a worker could lift comfortably over a substantial period of time under certain factors.
- LC is the load constant; the maximum weight limit in ideal situations. The remaining multipliers lower this limit according to factors that cause the situation to be less than ideal.
- HM is the horizontal multiplier, calculated based on the horizontal distance of the object to the worker (centered on the ankle bones).
- VM is the vertical multiplier, calculated based on the vertical distance from the object to the floor.
- DM is the distance multiplier, calculated based on how far the object travels vertically from the start to the end of the lift.

- AM is the asymmetric multiplier, calculated based on the angle of the line between the object and worker relative to the worker's sagittal plane. NIOSH recommends that asymmetric lifting be avoid in most circumstances.
- FM is the frequency multiplier, calculated based on the number of lifts per minute, the total duration of the lifting work, and the vertical position of the lifted objects.
- CM is the coupling multiplier, calculated based on how well the object can be handled. An object that is harder to grip or pick up will have a lower CM.

The multipliers of the RNLE can be set by consulting the tables in the RNLE applications manual. To determine if a given lift is safe, the Lifting Index can be calculated:

$$LI = \frac{Weight}{RWL}$$

$LI > 1$ indicates an unsafe lift, higher values increase this risk for lower back pain. While the LI is a useful metric for determining if lifts are unsafe for workers, it is impractical to calculate in a timely manner without rigorous setup and testing. Collecting all the required data and consulting the tables in the manual is not possible in a general, non-laboratory setting.

The American Conference of Governmental Industrial Hygienists (ACGIH) developed Threshold Limit Values (TLV) for lifting based on the RNLE, defining twelve zones relative to the worker's body at which various maximum lifting loads are defined (Figure 1). These zones can be simplified into groups of low, medium, and high-risk zones to provide more general guidelines for safe lifting that can be applied without careful measurement of lifting factors [6].

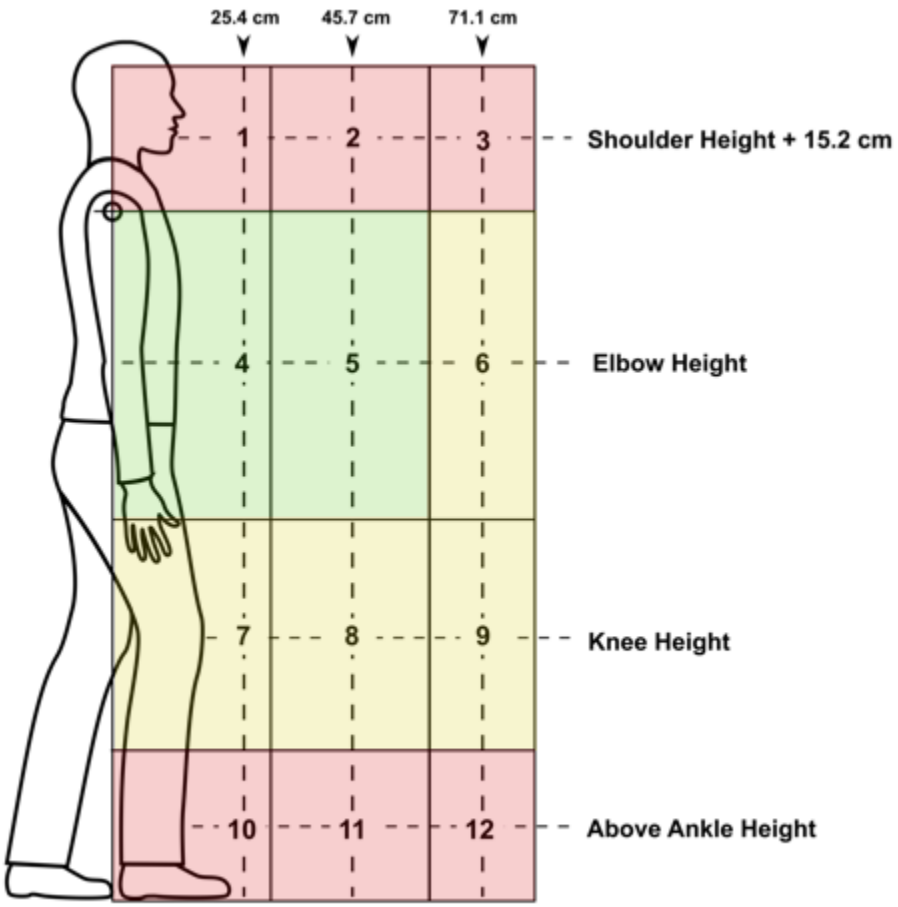


Figure 1: Lifting zones defined by the ACGIH Lifting TLV. They are divided into low-risk (green, 4-5), medium-risk (yellow, 6-9), and high-risk (red, 1-3 and 10-12).

In a previous study assessing an algorithm for predicting RNLE factors, NIOSH collected IMU data from 10 subjects performing lifting action. Each subject performed 6 lifts in each ACGIH zone, for a total of 720 lifting actions [7]. This dataset presents significant opportunities in advancing the state-of-the-art of HAR research, as very little work has been done with respect to specialized activities. With permission from NIOSH, this dataset was used as the main source of training and refinement for the models developed in this work, providing insight into the efficacy of machine learning applied to specialized activities for worker safety.

1.1.2 LSTM Neural Networks

HAR problems often involve identification of actions via an input of time-series data. Such input sequences pose problems for traditional learning strategies, as they are often unable to capture the temporal relations that define an action, such as the periodic sequence of walking. Recurrent neural networks (RNNs) are a form of neural network developed to handle data sequences as input. An RNN cell typically contains an internal state that is calculated using an input at a particular time step and the cell state of the previous time step. While RNNs allow for improved processing of time-series data, they are susceptible to the vanishing or exploding gradient problem, making training difficult [8]. To address these issues, Hochreiter and Schmidhuber introduced the Long Short-Term Memory (LSTM) cell [9].

1 This is defined mathematically as:

$$
\begin{aligned}
f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\hat{c} &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \hat{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Where:

- x is the input vector
- f, i, and o are the forget, input, and output gates respectively
- c is the cell state
- h is the hidden (or output) state
- W, U, and b are the weights and biases learned during the training process
- σ are activation functions

A graphical representation of an LSTM cell is shown below.

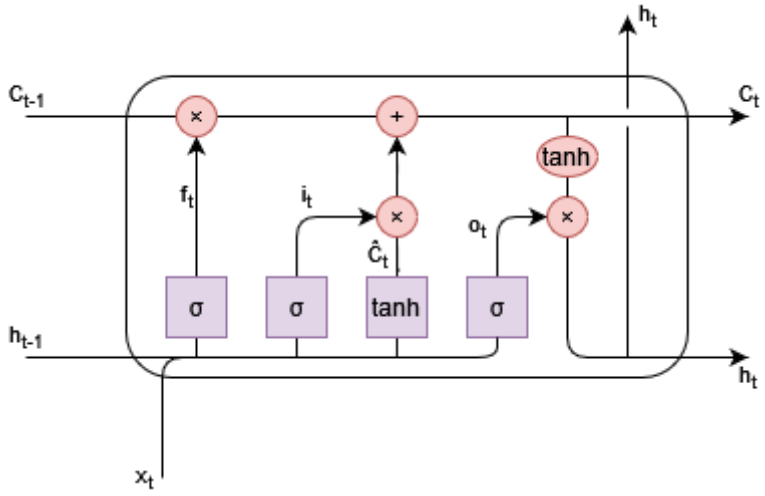


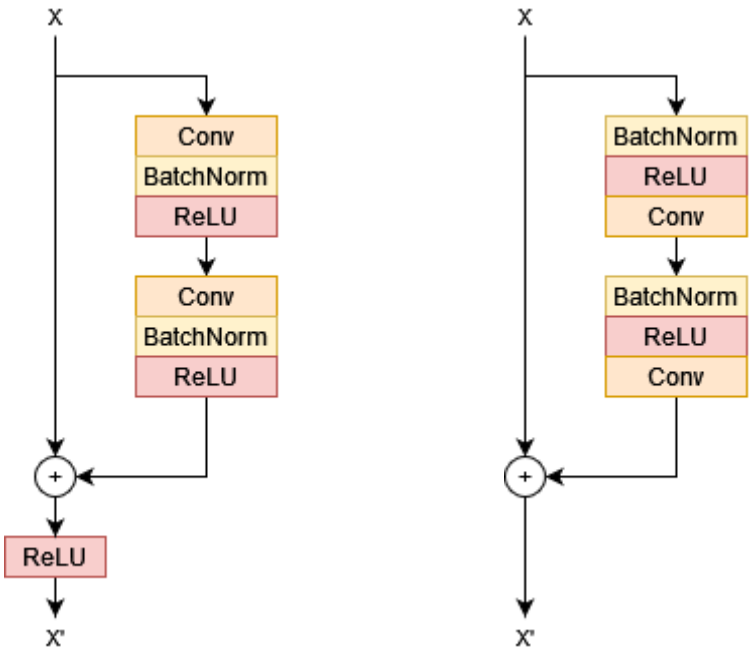
Figure 2: A diagram of a standard LSTM cell.

LSTM networks have seen success over traditional RNNs in many areas of sequence processing, including prediction, classification, and language processing, in a variety of domains [10, 11, 12]. Several advances to the structure of LSTM networks have been proposed, such as peephole LSTM [13] and convolutional LSTM [14], further cementing LSTM as a standard for sequence processing.

1.1.3 Residual Neural Networks

Convolutional neural networks (CNNs) have brought significant advancements to signal and image processing with their ability to capture special relations and extract features with significantly fewer parameters than fully-connected networks [15]. Despite their good performance, CNNs also suffer from the vanishing/exploding gradient problem, especially when more layers are added to the network, making it difficult to train a network capable of solving complex problems. To address these problems, He et al. introduced residual neural networks [16]. In a residual network, the residual function $F(x) = H(x) – x$ is learned rather than directly learning

the function H(x). This is formulated in the network as H(x) = F(x) + x, utilizing a 'skip-connection' where the input to the residual block is added to the output of the learned function F(x). The impetus for this approach is that in the case where the identity mapping is optimal for H(x) (as may likely be the case for a few layers in very deep networks) it is easier to push F(x) to zero rather than directly learning an identity mapping for the input. Residual networks have been shown to achieve greater accuracy in many recognition challenges and allow for significantly deeper networks to be constructed for more powerful representation. Further advances to the structure of the residual blocks, such as the order of activations have been introduced [17]. The architecture of a standard residual block and a residual block with pre-activation are shown in Figure 3.



(a) (b)

Figure 3: (a) A standard residual block operating on input X. A convolution layer is used here, but other layers are possible. BatchNorm refers to Batch Normalization, and ReLU refers to the activation function Rectified Linear Unit. (b) A pre-activation residual block. BatchNorm and activation functions are performed before the layer operation, rather than after.

1.2 Motivations

The primary motivation for this work is the lack of available research on HAR systems for specialized tasks, especially with regard to worker safety. Prior research is focused on widely applicable activities such as walking, standing, running, and sitting, of which there is an abundance of publicly available data [17, 18, 19]. There is significantly less research into recognition of specialized activities, likely due to the difficulty of collecting an expansive labeled dataset for a given activity. Specific activities such as lifting require time-consuming controlled trials to generate accurately labeled data, as opposed to more general activities that can be collected over the course of a subject's standard day. As a result, recognition of specialized activities is an underdeveloped area of research. The NIOSH lifting dataset used in this study [7] is several orders of magnitude smaller than publicly available datasets for general activities, which poses considerable challenges for constructing an accurate HAR system. Prior work on utilizing this dataset [20] achieved good performance using preprocessing strategies and specialized model architecture but did not address the problems with the size of the dataset directly at the data source.

Addressing worker safety concerns is also a primary motivator for this work. Back pain is a primary cause of occupation-induced disability, and most occupational back pain is caused by roles requiring repetitive tasks such as lifting of heavy objects [3]. Proper monitoring of risky behaviors can reduce risk and ensure that workers are not consistently required to perform unsafe actions as part of their regular duties, but common ergonomic risk assessment methods require a safety

professional to observe a worker's postures throughout a work shift [1]. This type of assessment is subjective and requires additional intensive labor. An automated system that can both detect behaviors and classify their risk levels can scale to many workers and provide real-time feedback to both reduce the risk of health complications for the workers and provide long-term safety data for supervisors.

HAR systems have been extensively studied in a variety of different sensor modalities [21], and have seen particular success in video-based activity recognition [22]. Unfortunately, video-based risk assessment would require the installation of cameras in every possible lift location for full coverage, which is prohibitively expensive and possibly invasive to worker privacy. Permanent installations are also impractical in temporary work sites such as construction zones. Sensor-based approaches, in contrast, can be distributed at low cost and are often already integrated in common devices such as smartphones and smart watches. A sensor-based approach to HAR could easily be implemented at scale to address safety concerns for workers before injuries occur.

1.3 Contributions

This work proposes convolutional LSTM neural networks capable of detecting when lifting action occurs and classifying the level of back pain risk the lift poses. The effectiveness of data augmentation for substantially small IMU datasets is also assessed, and augmentations are applied for increased performance. The proposed system is able to achieve good performance on a specialized dataset with limited data.

1.4 Organization

Chapter 2 presents an overview of previous works related to HAR systems. Chapter 3 introduced the dataset used in this work and describes the preparation of data for model

consumption. Chapter 4 presents both the detection and classification models and the training process used. Chapter 5 analyzes the results of the proposed system and the effects of data augmentation, and studies optimal sensor placements through saliency maps. Chapter 6 presents conclusions of the work and proposes areas for future research.

Chapter 2. Related Works

2.1. Introduction

This section provides an overview of recent research into the applications of machine learning to HAR problems. It then identifies areas for further improvement in current literature that are addressed by this work.

2.2. Previous Works

Chen and Xue [19] utilize a deep CNN to classify 8 typical activities such as walking, running, jumping, falling, etc. The dataset utilized was collected by the authors using the tri-axial accelerometer of an Android device. The kernel of the CNN was sized to span two out of three of the axes, to capture axial relationships of the data.

Snyder et al. [20] perform risk analysis of the NIOSH lifting dataset by reformatting the time-series IMU data as a 2D image that is filtered and then used as input to a CNN utilizing average pooling rather than the more standard max pooling. They achieve 90.6% accuracy in classifying the risk level of a lift.

Weiss et al. [18] introduced Actitracker, a smartphone-based activity monitoring system that provided activity reports to the user with nearly 90% accuracy based on user reports. Actitracker employed a universal model that became more personalized to the user after an initial training phase.

Ignatov [23] utilizes a shallow CNN for feature extraction and combines the extracted features with statistical features such as the mean and variance of the sequence for a final classification score via a fully-connected layer. They evaluate the model on two different datasets and perform a cross-dataset evaluation by training on one dataset and evaluating the model performance on the other.

Anguita et al. [24] introduce a dataset of daily living activities collected via smartphone and evaluate the performance of a support vector machine (SVM) used to classify the activities. The SVM achieves excellent performance on most classes but shows relatively worse performance on the sitting class, often misclassifying it as standing.

Ordoñez and Roggen [25] utilize stacked 1D convolutional and LSTM layers to perform automatic feature extraction and capture temporal relationships between these features, achieving state-of-the-art scores on the OPPORTUNITY [26] and Skoda [27] datasets.

Hammerla et al. [28] compare several different model architectures across three different datasets, including standard deep neural networks, CNNs, and RNNs including LSTM, achieving best results utilizing a bi-directional LSTM network.

Ravì et al. [29] introduce a model designed for low-power devices to enable real-time classification on wearable devices. Features are extracted from the spectral domain to reduce variation due to sensor placements or rotation.

Bhattacharya and Lane [30] utilize Restricted Boltzmann Machines (RBM) to design a HAR system implemented on a smartwatch. They show competitive performance on a range of classification tasks while utilizing an acceptable level of resource consumption for the smartwatch platform.

Finally, Guan and Ploetz [31] show that ensemble methods can improve performance in HAR scenarios. They utilize an ensemble of LSTM models and achieve performance improvements over several recent methods across multiple datasets.

2.3. Limitations of Previous Works

The majority of past works are developed and evaluated with common publicly available datasets, such as OPPORTUNITY and Skoda. While this is helpful for directly comparing model performance, it means that it is difficult to assess how well these models will perform in more specialized scenarios for which they were not designed. This represents a hole in current HAR research, where models are developed for common household tasks and not much else. There is very little research regarding utilizing machine learning for specialized tasks.

Additionally, the tasks that are commonly evaluated are often easily separable, even to an untrained observer. Previous works focus on tasks such as walking, running, and jumping, which are significantly different activities with their own IMU signatures. Few works focus on separating instances of different types of the *same* activity, as in the case of lift risk assessment. These models show a significant decrease in performance when classifying similar activities, such as standing vs. sitting or walking upstairs vs. walking downstairs. As there is so little difference between a safe and unsafe lift that an observer may not be able to tell the difference, current HAR models may be unable to adequately classify between the two. This work addresses the hole in current research by focusing on a specialized task and separating instances within that task. Therefore, the model is evaluated on how well it can determine which type of lift is occurring, although the action always consists of the same basic motion.

Chapter 3. Data

3.1. NIOSH Lifting Dataset

The IMU dataset used in this study was collected by NIOSH for use in a previous study [7]. Six IMU were placed on subjects' bodies and captured motion data while the subjects performed manual lifting actions at various levels of risk defined by the ACGIH TLV for Lifting, based on the zone the object was lifted from.

Each subject had IMU sensors placed on their dominant thigh, waist, upper back, dominant upper arm, and both wrists. Each sensor collected accelerometer and gyroscope data at a rate of 25 Hz. A given data sample consisted of a subject lifting a small wire frame, turning, walking a short distance, and placing the object down. Therefore, there is a large amount of data both before and after the lift. A total of ten subjects participated in the study; each subject performed six trials in each zone for a total of 720 lifting trials. Additionally, each subject performed 11 non-lifting trials such as standing, walking, and jumping. Figure 4 shows select sensor data for a single trial.

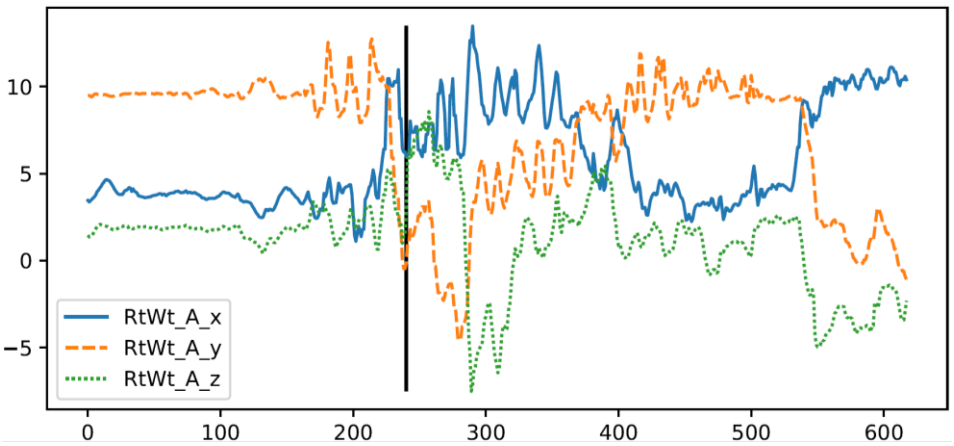


Figure 4: Accelerometer data for a single trial. Collected from the IMU sensor mounted on the subject's right wrist. The vertical black line marks the start of the lift action.

3.2. Data Preparation

The same dataset was used for both the detection and classification tasks. However, since the tasks have different input and output requirements, the data was prepared for model consumption in different ways.

3.2.1 Detection

The lift detection model must determine if a lift is occurring at a given moment in time. To facilitate this, each trial was segmented into samples using a sliding window with overlap. The label of a sample was determined in a probabilistic manner based on the proximity of the last time step in the segment to the start or end of the lift. For example, a segment that ends exactly at the start of the lift was given a hard label of 100% lift start. A segment that ends some number of frames before the lift was assigned a fuzzy label giving probabilities for both lift start and neutral. A visual depiction of these labels for each time step over an entire trial is shown in Figure 5.

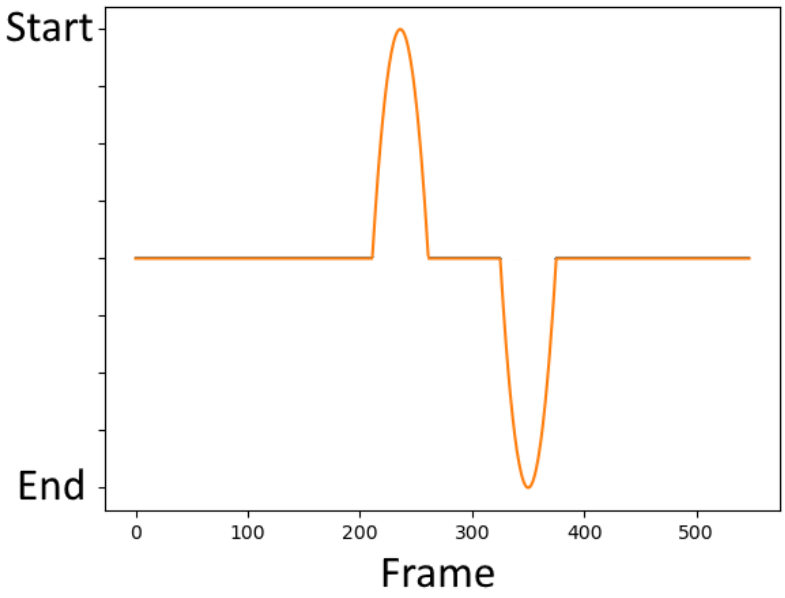


Figure 5: Start/End of lift labels for every frame in a single trial.

Labeling based on the last frame in the segment allowed the model to operate in an "on-line" fashion, where a prediction for a specific moment in time uses only information that is already available. For instance, the model can make a prediction at time t using time segment $[t - 24, t]$, and then continue on to predict the next time step $t + 1$ using $[t - 23, t + 1]$. In this way, the system can perform lift detection in real-time with minimal delay in feedback for the user. The model's predictions for each time step can also be assembled in order for a visual representation of lifts detected over time. The full pipeline of segmenting, predicting, and assembling is shown below.

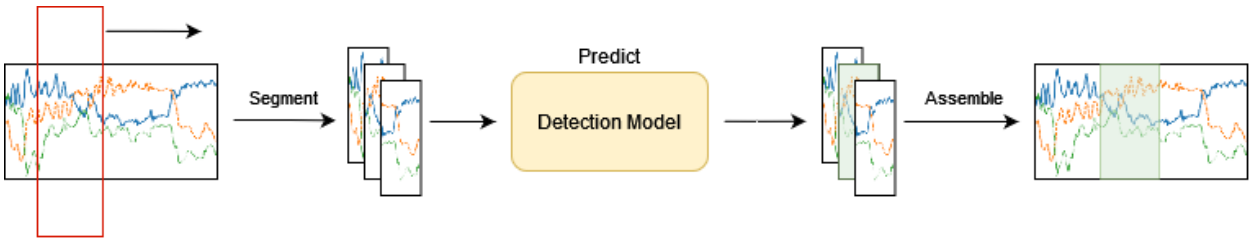


Figure 6: Lift detection pipeline for a single trial.

The final input to the detection model was a one-dimensional sequence of length 25 with 36 channels, one for each axis of a given sensor. Empirical tests showed that scaling of data reduced performance, so raw data were used with no scaling or preprocessing.

3.2.2 Classification

The input data for the classification model required no special formatting, as the entire trial sequence was used as input. This was done for two reasons. First, lifting action could occur anywhere within the sequence of data. Using the entire sequence allowed the model to learn to classify the lift regardless of where in the sequence it occurred, improving the model's translational invariance and reducing the need for precise detection of the lift before classification. Second, human movement is affected by the intent of the actor, and therefore there may be trainable data

in the motion both before and after the lift, such as changes in gait or posture [32]. Using the entire sequence ensures that this information is captured and used. Since the length of the lifting trials varied, each trial was zero-padded to the length of the longest trial, resulting in samples with a length of 750 frames and 36 channels. As above, raw data with no further preprocessing were used as input.

Chapter 4. Proposed Models

4.1. Introduction

Both models were developed using LSTM layers, as they have been shown to perform well on time-series data [9]. Convolutional layers are included to perform automatic feature extraction, replaced by residual blocks in deeper networks. The specifics of each model are detailed below.

4.2 Detection

The lift detection model consisted of a two convolutional layers with max pooling, two LSTM layers, and a fully-connected softmax output layer that classified the sample as a lift start, lift end, or neither. The activation function for convolutional layers was the rectified linear unit (ReLU), and for LSTM layers was the hyperbolic tangent function (tanh). Hyperbolic tangent was used for LSTM layers because the ReLU function often resulted in model divergence when used in those layers. The model architecture is shown in Table I.

Table I. Detection Model Architecture

Layer	Units	Activation
Conv1D	64	ReLU
MaxPool1D	-	-

Conv1D	128	ReLU
MaxPool1D	-	-
LSTM	128	tanh
LSTM	64	tanh
Dense	3	softmax

4.3 Classification

The architecture of the model was based on DeepConvLSTM, an LSTM-based activity recognition model consisting of convolutional layers acting as feature extractors followed by LSTM layers modeling temporal relationships between features. This combination allows for the model to work on raw IMU data without manual feature extraction, which helps the model generalize and minimize engineering bias [25].

The model architecture was modified from the standard DeepConvLSTM by replacing the convolutional layers with residual blocks. In this model, residual blocks with pre-activation were used [17], as they have been shown to improve regularization and ease optimization compared to traditional residual blocks. As in the original DeepConvLSTM formulation, no pooling was used after convolutions. The model was constructed using six residual blocks and two LSTM layers with 128 units each, with an additional fully-connected feedforward layer leading to a fully-connected softmax output with four neurons representing the probability of the sample as an instance of one of the three risk levels or non-lifting. The activation function used for LSTM layers was tanh. while the function for all other layers was ReLU. The full model architecture is depicted in Table II.

Table II. Classification Model Architecture

Layer	Units	Activation
Residual Block	128	ReLU
Residual Block	128	ReLU
Residual Block	128	ReLU
Residual Block	128	ReLU
Residual Block	128	ReLU
Residual Block	128	ReLU
LSTM	128	tanh
LSTM	128	tanh
Flatten	-	-
Dense	512	-
BatchNorm	-	ReLU
Dense	4	softmax

4.4 Training

Both models were implemented in the Python programming language using the TensorFlow 2 library. Training was performed on a machine running Ubuntu Linux with an NVIDIA GeForce RTX 2080 Ti for GPU acceleration, using a categorical cross-entropy loss function with Adam [33] as the gradient descent algorithm. To ensure the models were fully trained, early stopping was used rather than train for a specific number of epochs, and various cross-validation techniques were employed to reduce variation of results.

4.4.1 Early Stopping

A significant problem that arises when using neural networks is that of overfitting. Overfitting occurs when a model has too many parameters or uses irrelevant features for its predictions, reducing loss on the training set but simultaneously reducing generalization capabilities [34]. Since it is difficult to know ahead of time how many parameters a model should have or what features to use for a given problem, these parameters cannot be directly adjusted. In practice, a validation dataset is used to assess the model's performance during the training and tuning process. While this provides a standard assessment of model performance after training, it is still hard to tell how long the model should be trained for to achieve the best generalization capabilities, since lower training loss does not correlate with generalization due to overfitting.

Early stopping [35] is a regularization technique that allows for models to train for the optimal amount of time while preventing overfitting. Rather than train for a specific number of epochs, the training process continues up to a maximum number of epochs while loss on the validation dataset is monitored every number of training rounds. If the validation loss stops consistently decreasing (for example, it hasn't decreased in the last number of epochs), then the training is considered complete and is stopped early. This ensures that the validation loss never

increases, which is a clear sign of overfitting and reduced regularization capabilities. After training is complete, a recent set of weights that achieved the lowest validation loss can be kept and used as the final model weights, ensuring the best chance at a model that performs well on the test data.

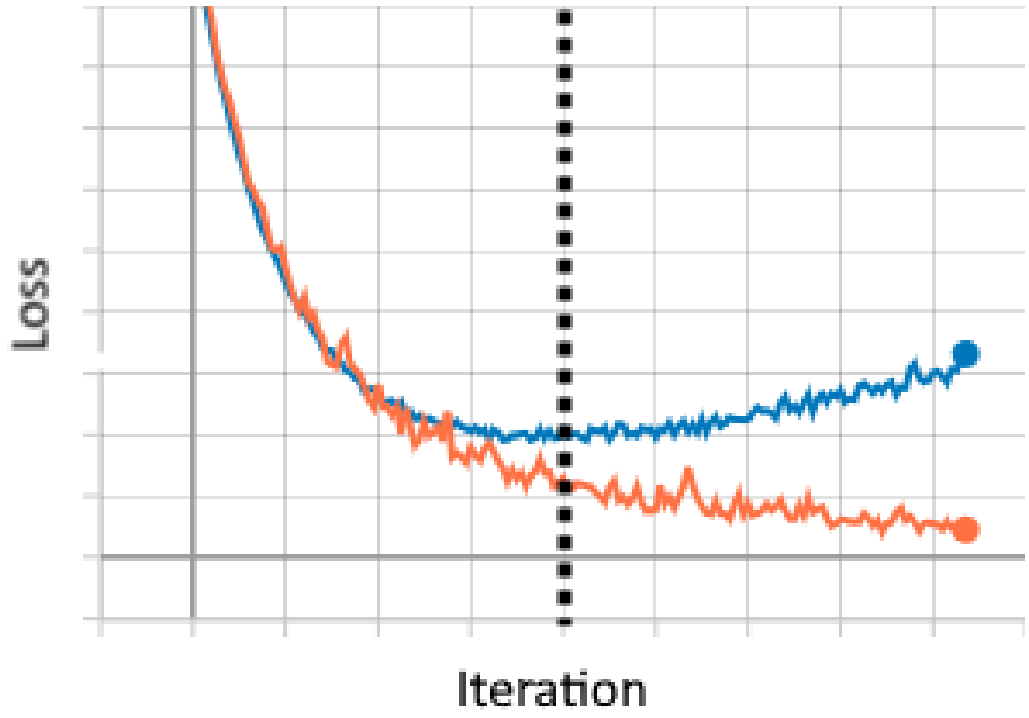


Figure 7: Sample curves for training (orange) and validation (blue) loss during training. The dotted vertical line denotes the approximate iteration where validation loss stops decreasing and overfitting begins, where early stopping would trigger.

For this research, training was halted if there was no improvement in validation loss after 50 epochs, up to 500 epochs. After training, the weights corresponding to the lowest validation loss from the last 50 epochs were kept. This ensured that the models were trained completely, while preventing them from training too long and overfitting. In practice, the classification model tended to train for around 250 epochs before stopping, while the detection model trained for around 120 epochs before stopping.

4.4.2 Cross-Validation

Cross-validation (CV) [36] is often employed to assess model performance and select models for deployment. A good CV strategy assesses model performance with little variation, so that the assessment can be assumed to be a reasonable approximation of the model's performance on real data. Two CV strategies are considered in this work: *k*-fold CV and leave-one-subject-out (LOSO) CV.

4.4.2.2 k-fold

k-fold cross-validation is a common strategy for low-variance model assessment [36]. In *k*-fold CV, the dataset is split into *k* different train-validation folds, where the validation data in each fold is different and spans the entire dataset across all folds (Figure 8). A separate model is then trained and evaluated using each fold, and these evaluations are averaged for the final model evaluation. This generates an evaluation metric that spans the entire dataset while still only evaluating on unseen data.

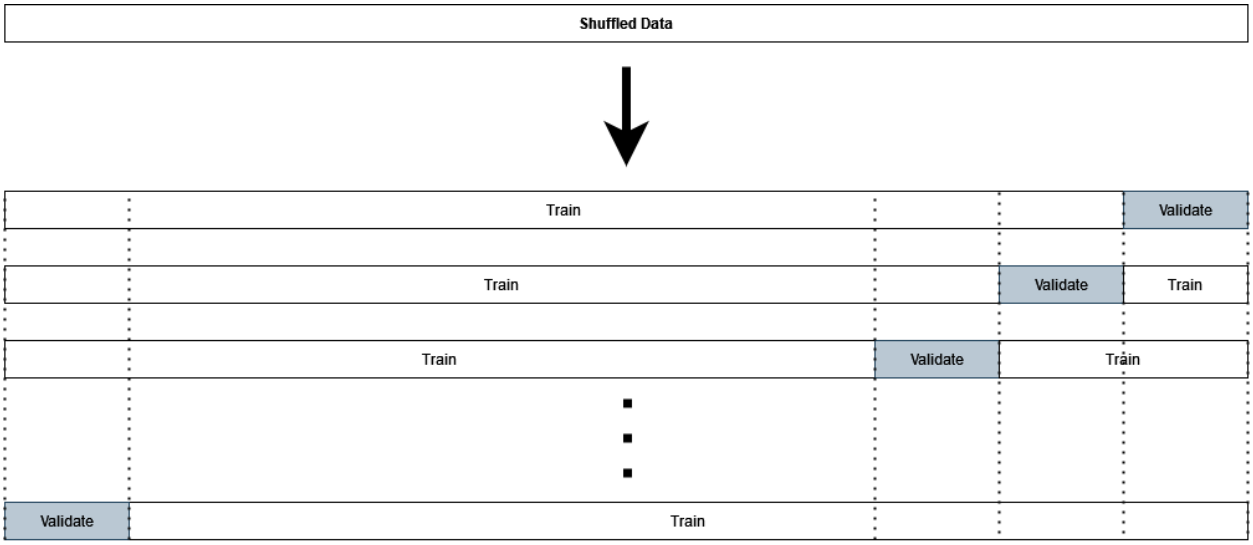


Figure 8: Creation of training folds for k-fold CV. Validation data are shuffled randomly and selected so that each sample is in a validation set once across k folds.

In this work, the number of folds was set as k=10. Ten separate train-validation splits were used to train 10 different models, and results were averaged for the final evaluation as above.

K-fold CV may be susceptible to subject bias; that is, models may use subject-specific data to learn, which will not transfer to new subjects. Since the validation data in k-fold CV contains data from subjects used in the training data, the cross-validation will not account for this shortcut during the evaluation, so the reported metrics may overestimate model performance [37]. Leave-one-subject-out CV does account for this, but timestamp labeling differences between different subjects in the NIOSH lifting dataset meant that LOSO CV could not be used for lift detection, so k-fold was used as a backup strategy. Since the action to lift a box with both hands is generally similar across populations, the risk of subject bias is likely to be low.

4.4.2.1 Leave-one-subject-out

Leave-one-subject-out CV is an improvement over k-fold CV that accounts for subject bias by simulating the model's performance on an entirely new unseen subject [37]. Data folds are constructed similarly to k-fold, but the validation set for a given fold consists of the entirety of a single subject's data, and the rest of the subjects' data are used for training. A fold is constructed and a model trained for each subject in the dataset, and evaluation metrics are aggregated as in k-fold CV.

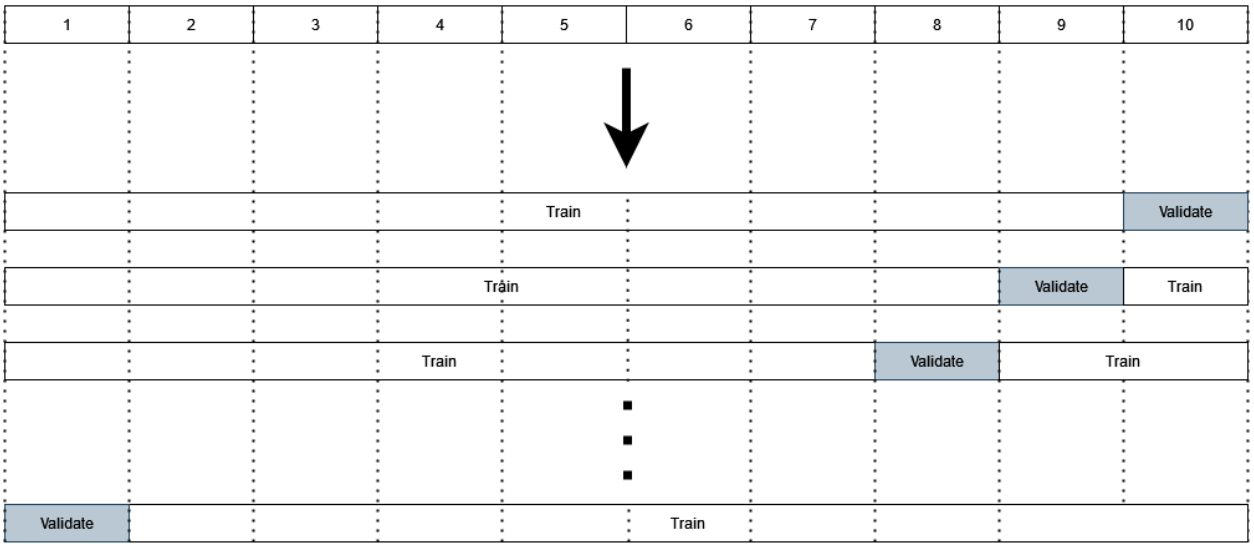


Figure 9: Creation of training folds for LOSO CV. Rather than choose a validation set of shuffled data, each fold uses all samples from a single subject.

Since each model is evaluated on a subject not present in the training set, there is no chance of subject bias. This approach assumes similar distributions of data and labels across subjects, as a model would perform poorly on a subject significantly different from the majority. As the NIOSH dataset was collected in a laboratory setting with the same trials across every subject, this was not a concern for this research (except for the lift start labeling differences as noted above). LOSO CV was employed to evaluate the classification model's performance.

4.5 Hyperparameter tuning

Hyperparameters are values that must be set prior to training and exist independent of the dataset. Correctly tuning these parameters is important for optimal model performance, as they can affect training and testing results as easily as the model architecture itself [38]. Three different hyperparameters were tuned during development of these models: learning rate (α), dropout percentage, and an L2 regularization parameter (λ). Different combinations of these three parameters using common values for each were tested using a grid search over common values for

each, and validation loss and accuracy were monitored to see which combination resulted in smoothest training and good performance.

4.5.1 Dropout

The dropout percentage is the portion of weights that are removed randomly omitted during a given training step, preventing the visible weights from using the hidden weights to co-learn complex features that may not generalize to the test data. This helps prevent overfitting by making the model explore more possible solutions and not get stuck in a local minimum by functioning as a form of model averaging [39]. Higher dropout percentages can reduce overfitting more, but may also cause instability in training, so this parameter must be tuned to find a good balance. Dropout was used in the classification model only.

4.5.2 Learning Rate

The formula for training a model f via gradient descent is defined as:

$$\theta = \theta - \alpha \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(x_i), y_i)$$

Where:

- θ are the model's parameters
- L is a loss function
- y_i is the true label of the sample x_i
- α is the learning rate

The learning rate therefore determines the amount of change to the model's parameters during each step of gradient descent. Lower values result in more stable training as the model's solution traverses the curve of the loss function more slowly and is less likely to miss the target minimum,

but also extend the training time and may cause the model to get stuck in a small local minimum. In contrast, higher learning rates result in faster training but may cause the model to overshoot the target minimum, so training is less stable.

4.5.3 L2 Regularization

L2 regularization adds a loss metric to the complexity of a layer, causing the model to keep its complexity low and reducing the chance of overfitting [40]. The regularization term is defined as:

$$L_2 = \lambda \|w\|_2^2 = \lambda \sum_{i=0}^{n} w_i^2$$

Where:

- w is the weights of a layer
- λ is the regularization parameter that controls the strength of the regularization.

The minimization of the λ term trends the weights of the layer towards smaller values in a distribution centered around zero. This encourages the model to learn simpler solutions with smaller weights, which are less likely to be overfit on the training data.

Regularization was included in each convolutional and dense layer of the models. The regularization parameter determines the importance of regularization loss compared to the standard loss. Higher values enforce simpler models, which can reduce overfitting but may lead to a model that is not complex enough to learn well.

4.5.4 Hyperparameter Results

Hyperparameter values were chosen by comparing the means and variances of the F1 scores achieved by the model across all folds of data. Values resulting in higher scores with lower variance are preferred, as the model is more likely to consistently perform well on unseen data. Results of the grid search over α and λ for the detection model are shown below.

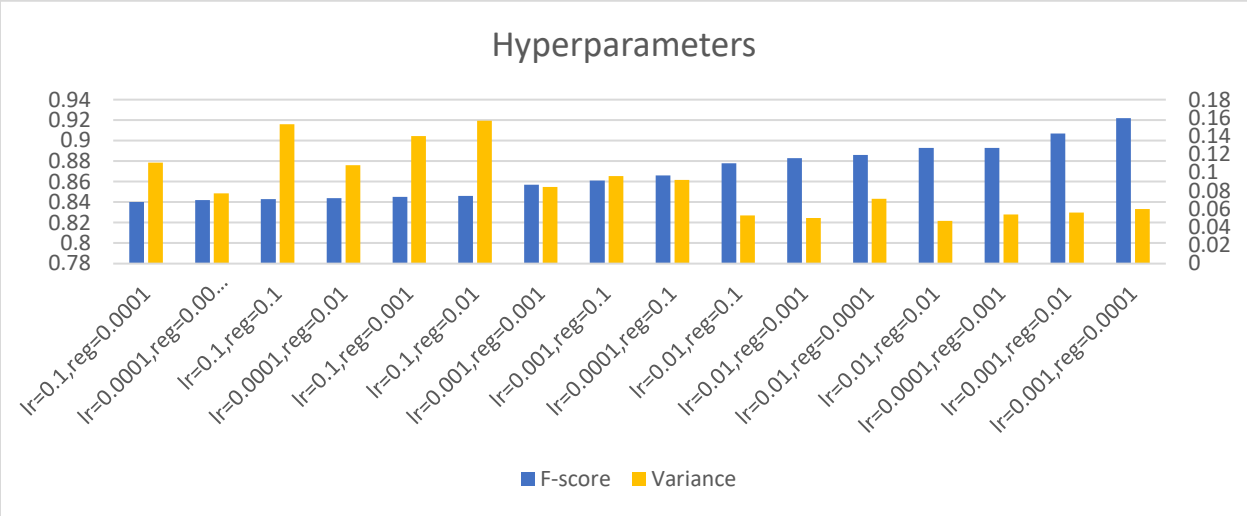


Figure 10: Results attained by the detection model parameterized by various hyperparameter values.

The results of the grid search showed that $\alpha = 0.001$ and $\lambda = 0.0001$ achieved the best results for the lift detection model. The same values achieved best results for the classification model as well, with a dropout parameter of 0.5.

Chapter 5. Results and Discussion

5.1. Proposed Model Results

5.1.1. Detection

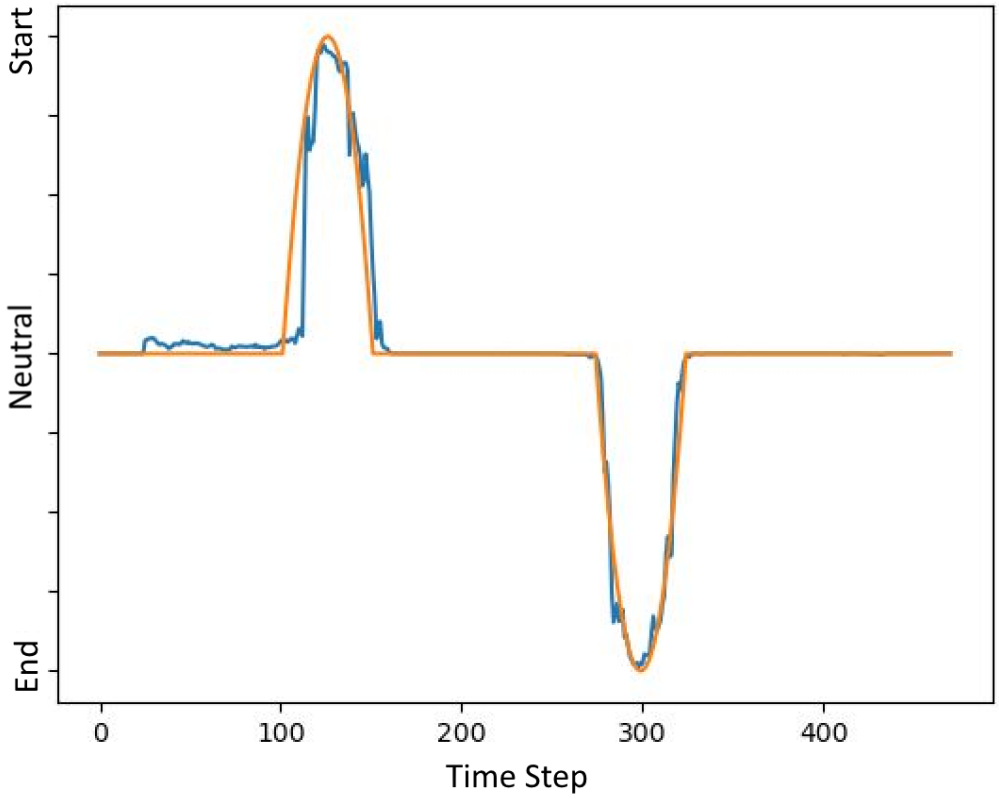


Figure 11: Model predictions (blue) for each time step in order, compared to the target labels (orange)

Figure 11 shows a sample output from the detection model over an entire trial sequence. For results analysis, the start or end time step of a sequence is marked the first time the prediction for a time step passes a threshold of 0.5 for the respective class. This mark is compared to the ground truth lift start or end to determine the error of the prediction in seconds. A binary analysis of whether or not a lift was detected in each trial sequence was also performed. The F-score for the model was

calculated using only samples where a lift was correctly detected within two seconds of the ground truth as true positive samples. Metrics for the model are given in the table below.

Table III: Lift Detection Results

Metric	**Result**
F score	0.980 ± 0.009
Start Time Error	0.861 ± 0.942
End Time Error	0.963 ± 1.244

The model is able to detect both the start and end of a lift with less than one second of error on average and achieves a high F-score for detecting the lift in a given sequence. In some cases, false positives are detected several times on non-lift data such as walking, as shown in Figure 12. Luckily, these false positives are rare, and can often be detected and accounted for by the classification model, as discussed in the next section.

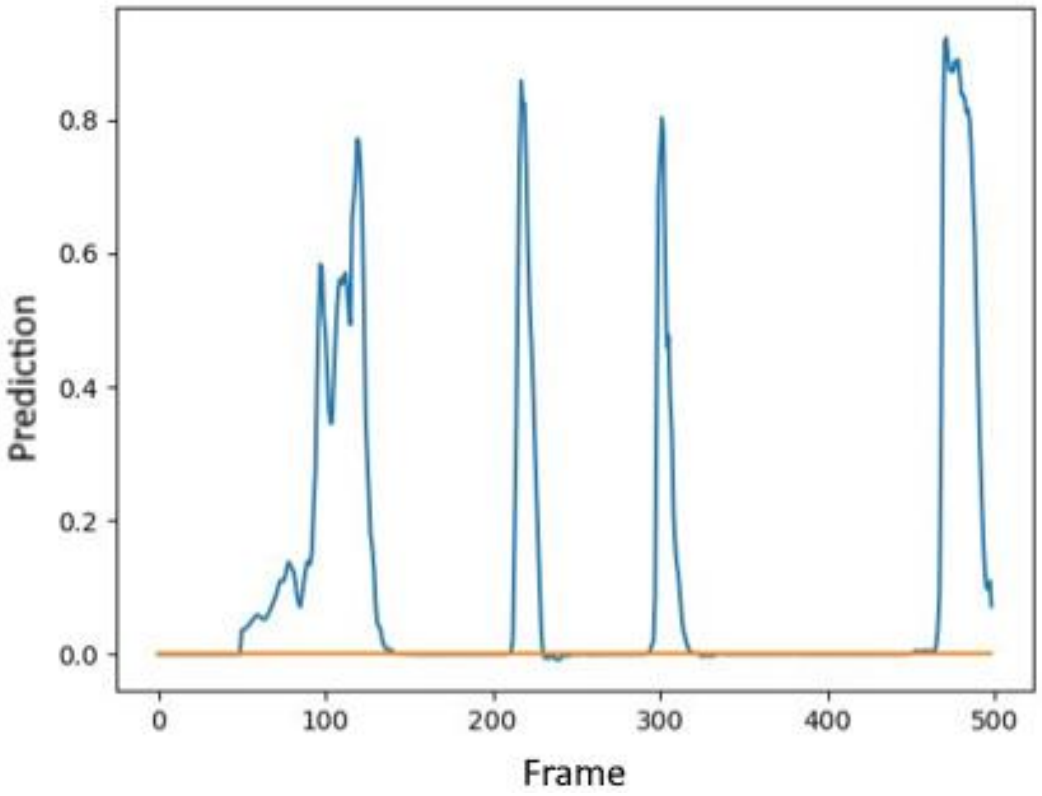


Figure 12: Model predictions (blue) for a series of samples from a 'Walk' trial.

5.1.2 Classification

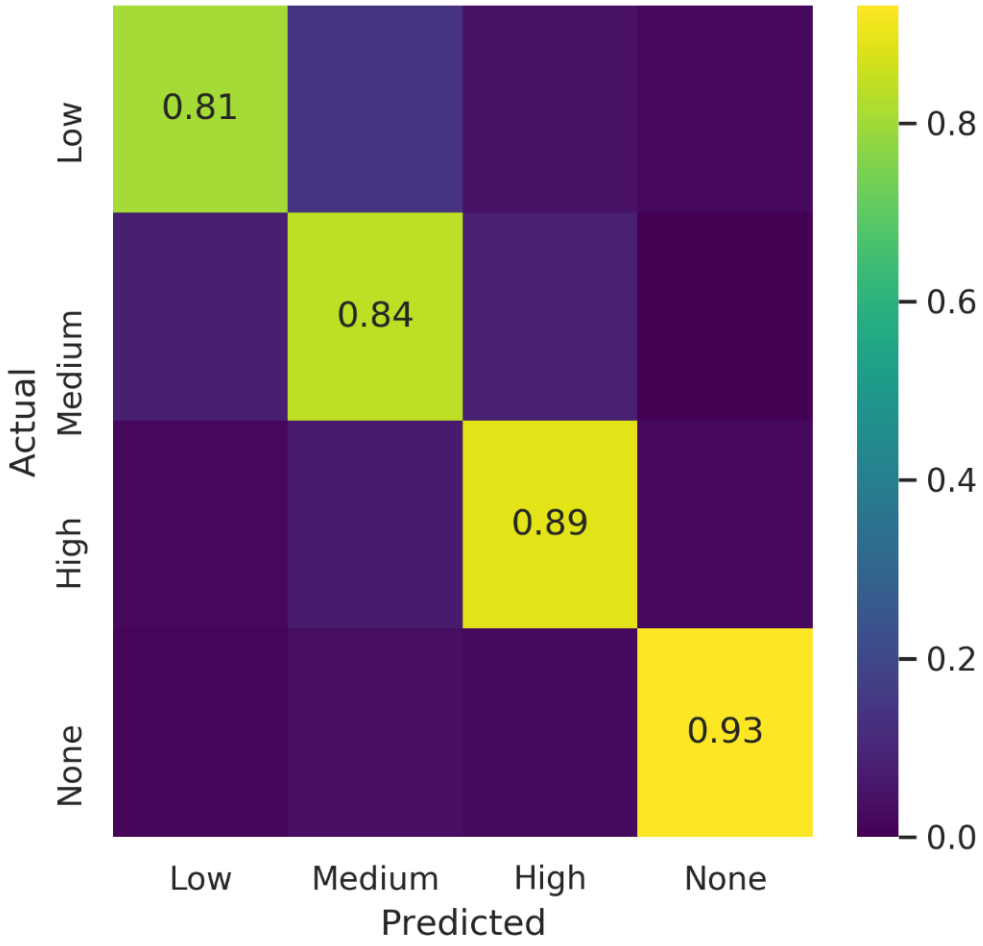


Figure 13: Confusion matrix showing model performance per-class. Results are normalized by row, so each value corresponds to the recall achieved for that class.

Figure 13 shows a confusion matrix for the classification model's predictions compared to the ground truth labels. Table IV compares the performance of the proposed model to other baseline models [19, 23, 25], as well as a standard linear kernel support vector machine. For each risk level, the F-score is calculated based on model predictions in or out of the class. For overall model assessment, a balanced accuracy score [41] is used to account for imbalance in classes in

the overall assessment by weighting scores in a given class according to its prevalence in the dataset. The proposed model shows a significant improvement for overall balanced accuracy over the next highest baseline, and achieves good scores over the baselines for each individual class as well.

The model appears to have a bias towards classes that are more prevalent, a common occurrence with imbalanced datasets [42]. In this case, lifts that are more risky are overrepresented in the dataset, simply because there are more lift zones that are high risk than medium risk, and more medium risk zones than low risk. This causes the model to be more likely to predict high-risk lifts, because it has more examples of them and achieves better scores in the overall dataset by preferring high-risk lifts. Although biased models show worse performance on classes it is biased against (thus lowering the overall balanced accuracy score), in this case a bias towards higher-risk lifts is acceptable. The cost of misclassifying a high-risk lift as lower-risk is much higher than vice versa, since missing multiple high-risk lifts would result in back pain risks not being addressed, increasing risk of injury. Therefore, increasing recall on high-risk samples should be a priority, even at the expense of precision with regard to false positives on low-risk samples.

The model performs very well at classifying non-lifting events. The long sequence of data used as input gives the model plenty of opportunity to examine long-term activities such as walking or running, and determine if a short-term lift occurs anywhere in the sequence. This acts as a sort of filter or fallback for the detection model, lessening the impact of false positives generated by that model.

Table IV. Proposed Model Comparison

Measure	SVM	Chen et al.	Ignatov	DeepConvLSTM	Proposed
Low F-Score	0.500	0.503	0.545	0.519	0.861
Medium F-Score	0.599	0.612	0.659	0.683	0.875
High F-Score	0.695	0.818	0.845	0.873	0.955
None F-Score	0.779	0.747	0.912	0.911	0.950
Overall Balanced Accuracy	0.555	0.558	0.643	0.660	0.886

5.2 Data Augmentation

Data augmentation is a common strategy for improving model performance on datasets of limited size, especially in the image domain [43]. Augmentation works by modifying data samples in ways that do not change the perceived label of the sample, such as rotation and scaling for images (an upside-down cat is still a cat, and should be classified as a such by an image recognition model). This allows for more generalization capability by simulating more types of samples the model may encounter, even if they are not originally represented in the dataset.

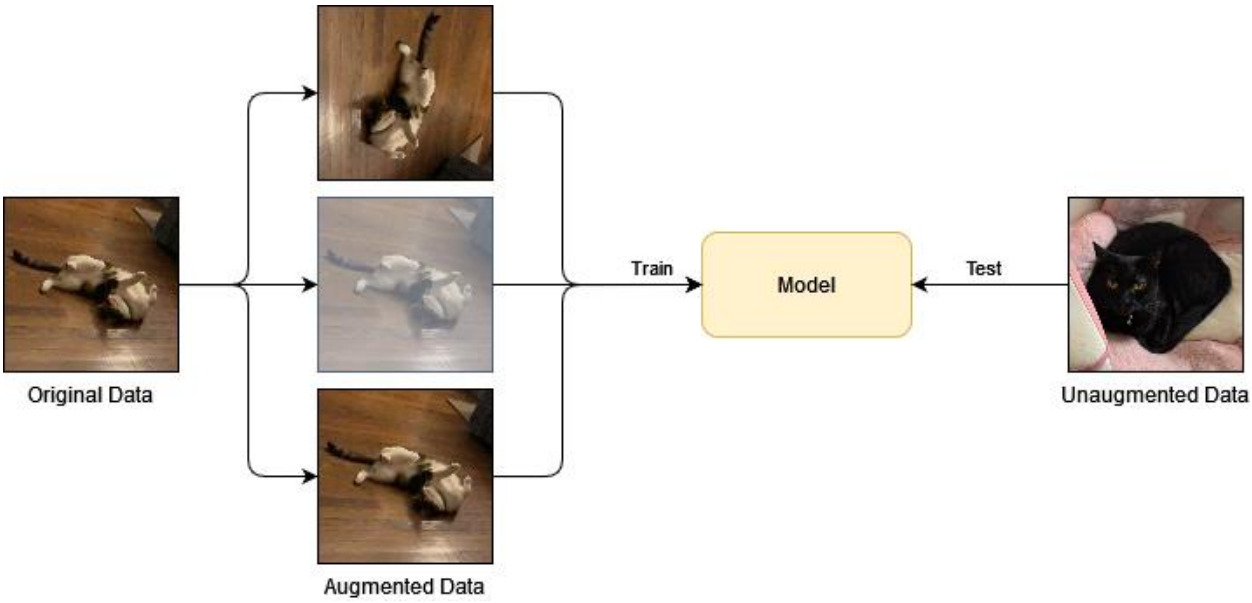


Figure 14: A standard data augmentation pipeline. The training data are subject to various transformations, and the model is trained on both the original data and the augmented data. The test data remains unaugmented.

A significant problem with applying data augmentation in the IMU domain is that it is difficult to determine what augmentations and how much of them can be applied without modifying the original label of the sample. For example, adding noise can allow for the model to be robust to small variations, but too much noise can obscure important features of the data. For images, a visual inspection is often enough to tell if an image has been modified too much as to be unrecognizable. This type of subjective analysis is not possible in the IMU domain, so certain transformations may destroy discriminative features of the data and render the model unable to classify well [44].

5.2.1 Label-Preserving Augmentation

To determine if a transformation can be applied to augment the dataset without removing discriminative features, a technique known as Label-Preserving (LP) Data Augmentation [45] can

be applied. LP augmentation systematically determines at what level to apply an augmentation to prevent reduction in model performance via augmentations that change the label of a sample. The process can be broadly broken down into two steps:

1. Train a model on non-augmented data to establish a baseline model, keeping a validation set held out of training.
2. Apply augmentations with varying parameters to the validation data and test the baseline model. Use the model's performance on the augmented data to determine optimal parameters.

A pipeline for the second step is shown in Figure 15.

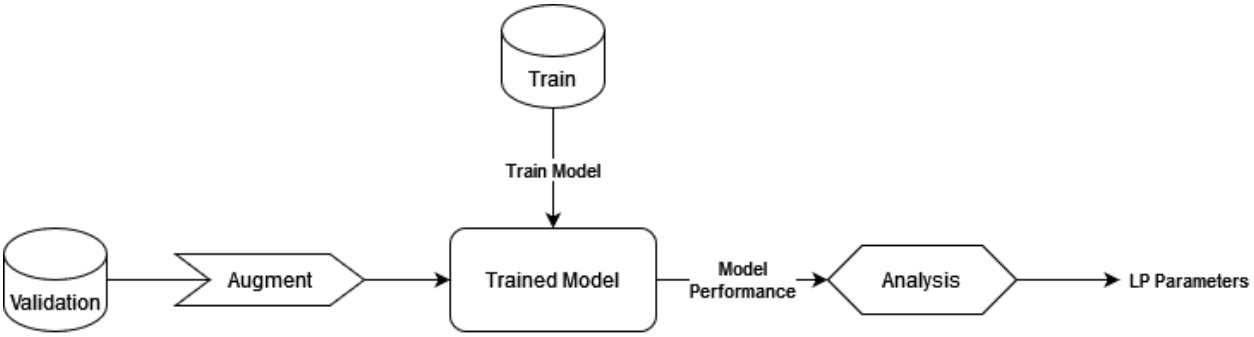


Figure 15: Label-Preserving Augmentation process

A graph showing the baseline model's performance according to the augmentation's parameters can be plotted to analyze how the augmentation affects performance and select optimal parameters. In general, performance of the baseline model will decrease with stronger parameters, as the samples move further from their original distribution. If there is a point at which the performance suddenly drops, the augmentation parameters are selected at that point. Otherwise, parameters are selected at the point where the model performance drops to 90% of the baseline. These parameters can then be used to apply augmentations to the training data.

LP augmentation was performed with both the detection and classification models to determine optimal augmentation strategies and parameters for augmenting the data. The discovered parameters were used to apply augmentations to the training data to determine if data augmentation is a valid strategy for improving model performance on small, specialized datasets, and how much improvement can be observed.

5.2.2 Augmentation Strategies

There are many possible transformations that can be applied to augment data. Common image augmentations include rotating, scaling, cropping, and adding noise [43]. Many of these are not suitable for IMU data. Instead, augmentations that reflect realistic data variances should be used. For example, a lift can be performed at different speeds while incorporating the same motions, so an augmentation that modifies the amplitude of acceleration signals could be used to simulate this. The augmentation strategies described in [46] are applied:

1. *Permutation* slices each sequence into multiple windows and then permutes the order of the windows. The tested parameter for this augmentation determines the number of windows.
2. *Time-warping* distorts the temporal dimension by deleting or interpolating each time step at varying levels of intensity, simulating slowing down or speeding up the signal at varying rates, parameterized by a smooth curve centered around 1. The tested parameter for this augmentation determines the variance of this curve.
3. *Scaling* multiplies each time step by random noise, randomly scaling the magnitude by a certain amount. The tested parameter for this augmentation determines the variance of the noise.

4. *Magnitude-warping* is similar to scaling, but the multiplier for each time step is determined by a smooth curve centered around 1, as in time-warping. The tested parameter for this augmentation determines the variance of the curve.
5. *Jitter* adds random noise to each time step. The tested parameter for this augmentation determines the variance of the noise.

A visual representation of these transformations on a data sequence is shown below.

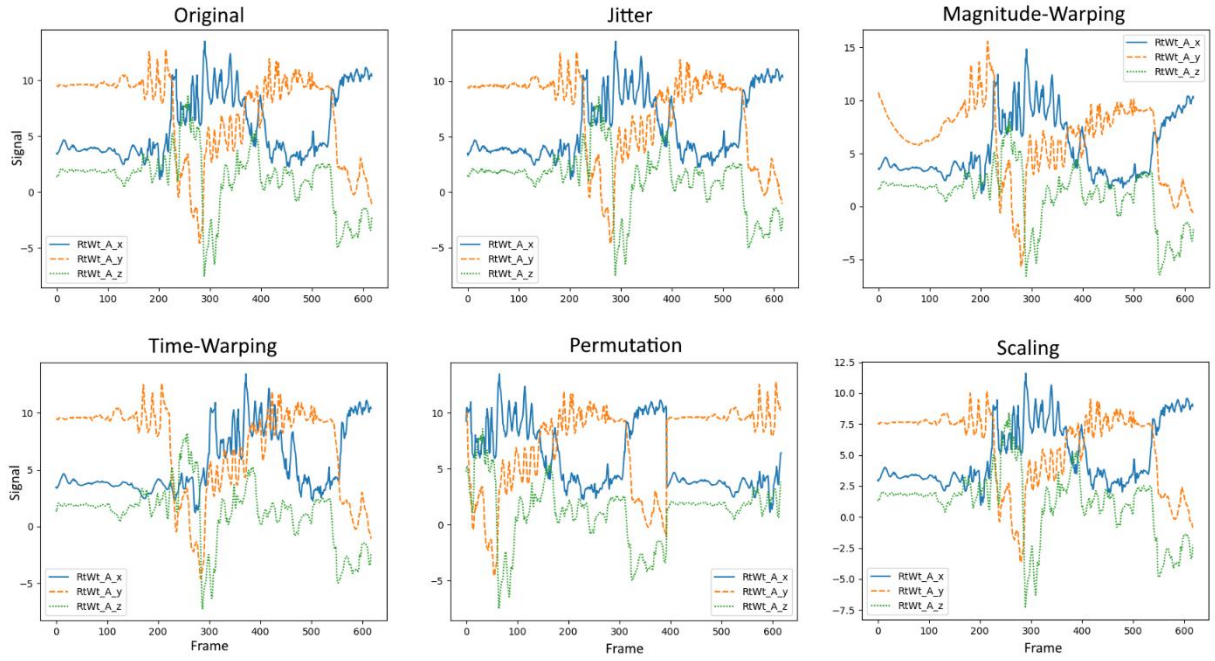


Figure 16: Effects of the augmentation strategies used in this work on right wrist accelerometer data from one trial.

5.2.3 Augmentation Results

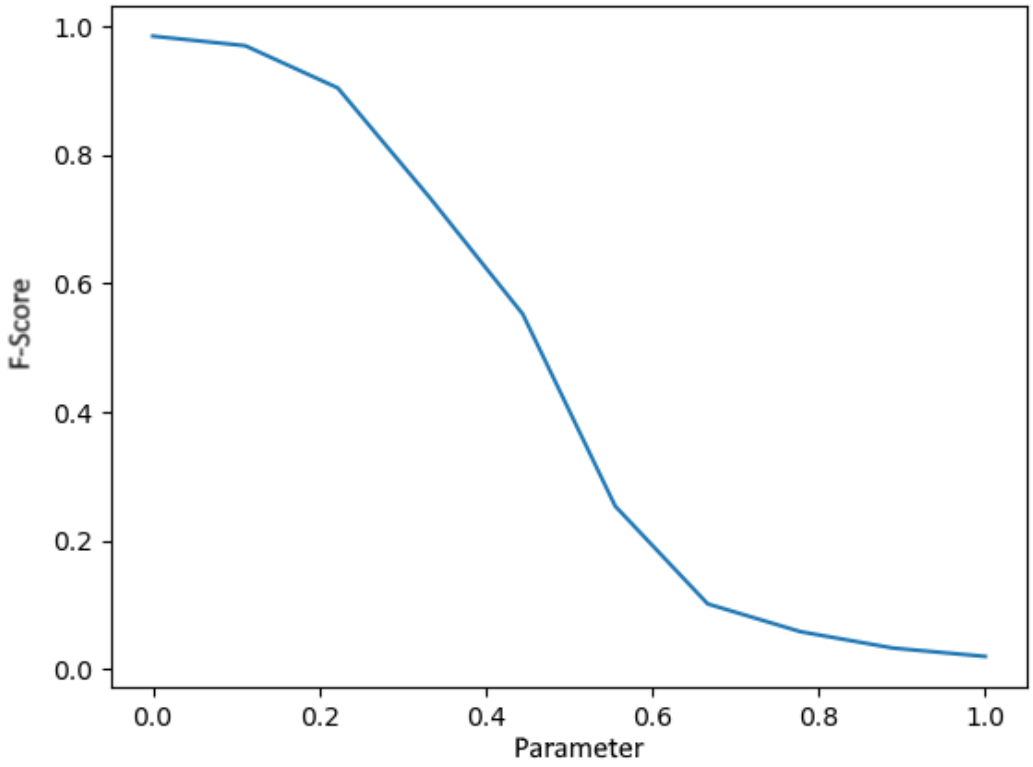


Figure 17: F-score vs augmentation strength for scaling of lift detection data.

The results of the LP augmentation for scaling can be seen in Figure 17. There is no sudden drop in accuracy on the augmented validation data, instead, there is a smooth curve with gradually decreasing performance. The scaling strength is set to 0.3, corresponding to where the baseline model achieved an F-score of 0.88, 90 percent of the baseline 0.98. The parameters for each transformation determined separately for detection and classification using LP augmentation are listed in Tables V and VI, as well as the f-score (for detection) or balanced accuracy (for classification) achieved by the model when training data was augmented using these derived parameters. Transformations were separately applied to each sample in the dataset once, resulting in a doubling of available training data. For example, the scaling augmentation was applied to each sample in the training dataset, and then the model was trained on both the original training data

and the scaled training data. This was repeated for each transformation to assess its applicability as a data augmentation strategy.

Table V. Detection Label-Preserving Augmentation Results

Transformation	LP Parameter	F-Score
None	-	0.980
Permutation	4	0.975
Time-warping	0.80	0.970
Scaling	0.30	0.975
Magnitude-warping	0.30	0.973
Jitter	0.80	0.975

None of the tested transformations worked well as a data augmentation strategy for the detection model. Although the first stage of the LP augmentation process worked as expected, the model performs the same or slightly worse when trained on a dataset of augmented training data, suggesting that data augmentation is not a good strategy for improving model performance in this case. This may be due to the size and number of samples generated by the segmenting of the data during preprocessing. Since there are a large number of small samples with limited information, the augmentation does not produce much more information the model can use, and the dataset is already large enough in this case for the model to learn.

Table VI. Classification Label-Preserving Augmentation Results

Transformation	LP Parameter	Balanced Accuracy
None	-	0.866
Permutation	2	0.836
Time-warping	0.83	0.849
Scaling	0.30	0.928
Magnitude-warping	0.35	0.920
Jitter	0.50	0.907

The LP augmentation process had much better results for the classification model, except for permutation. The permutation transformation resulted in an immediate drop in baseline accuracy during the LP process, suggesting that permutation is not a label-preserving transformation for this dataset under any parameters. For completeness, the model was tested on data augmented with a permutation of two segments, and as expected, performance dropped considerably. All other augmentations aside from time-warping result in an increase in model performance, up to 4 percentage points when scaling is applied. Applying more than one augmentation at a time to further increase the size of the data did not result in further performance increases. This suggests that data augmentation is a viable strategy in cases where there are a limited number of data samples, but each sample contains a large amount of information. The augmentation can use the available information to simulate a larger variety of samples and allow the model to generalize more easily.

5.3 Sensor Requirements

The presented solution utilizes the entire NIOSH lifting dataset, consisting of six IMU sensors, in the interest of completeness and accuracy. However, requiring a worker to be fitted with six sensors each time they are expected to work is unrealistic and would be cumbersome, interfering with important tasks and possibly adding additional noise to sensor readings. Reducing the required number of sensors would increase system utility and make it more likely to be accepted in a workplace. Since certain sensor placements may be optimal for a given HAR task [47], it is important to determine which placements can be safely removed from the available data with minimal adverse effect on performance. The important sensors for the lifting detection and assessment task were determined using saliency mapping [48], and additional evaluations were run with only these sensors including to test how well these models work with less available sensors.

5.3.1 Saliency Mapping

To determine which sensors are most important for recognizing lifting action, a technique known as saliency mapping [48] was applied. Saliency mapping ranks the importance of features for a given input by calculating the gradient of the output scores with respect to the input. These gradients show which input features result in a large change in the output scores given a small change in the input, and can be easily visualized for low-dimensional input in the form of heatmaps.

A simple example of a saliency calculation can be shown with a linear model for a class c:

$$S_c(I) = w_c^T I + b_c$$

Where:

- S_c is the output score of the model.
- I is the input to the model.
- w, b are the weights and bias of the model.

Since the input features are multiplied directly by the weights in the calculation of the score, features with higher weight associated with them have more effect on the output score. In this instance, it is clear to see that the magnitude of the weights directly corresponds with importance of the input features.

As a neural network is non-linear, the previous explanation cannot be applied directly. However, a linear approximation of a non-linear S_c can be calculated near a specific input I_0 using the first-order Taylor expansion:

$$P(x) = S_c(I_0) + S_c'(I_0)I - S_c'(I_0) * I_0$$

$$P(x) = S_c'(I_0) * I + S_c(I_0) - S_c'(I_0) * I_0$$

$$S_c(I) \approx w^T I + b$$

Where:

- $w = \frac{\partial S_c}{\partial I}\Big|_{I_0}$, the derivative of S_c with respect to I evaluated at a specific input I_0.
- $b = S_c(I_0) - wI_0$, a constant.

So, the derivative of the output scores with respect to the input correspond to the weights of this approximate linear function, and thus measure the relative effect of that input's features on the output score. This also makes intuitive sense, as the derivative of the model's output with respect to the input indicates how much a change in the input will change the output: larger magnitudes indicate larger changes in the output, and thus more importance for the input feature.

The saliency mapping process was performed with the classification model across all input features and output classes, utilizing the keras-vis library. Saliency maps for each input were averaged to create a "global" saliency map that captures the average importance of the input features for a given class across the entire dataset.

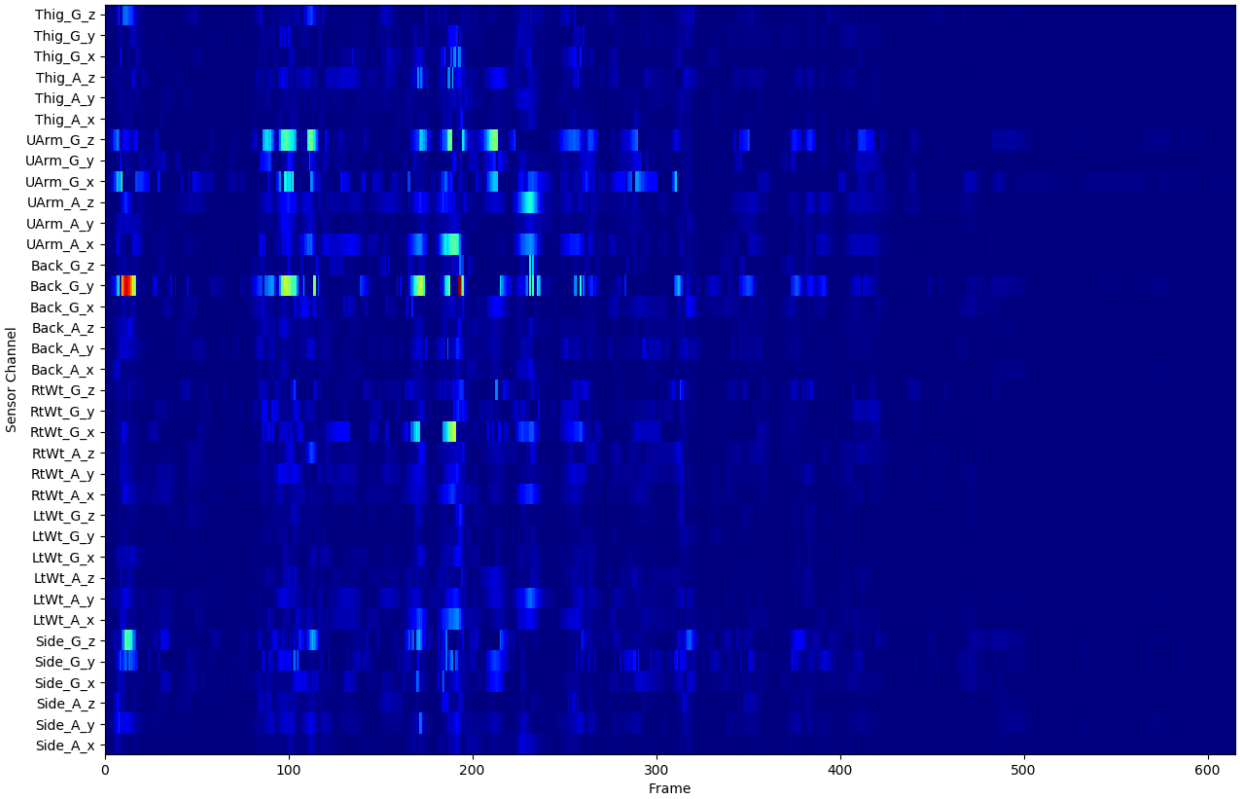


Figure 18: Average saliency map for the high-risk class. Warmer colors indicate larger gradients and thus higher importance for a given sensor channel at that time step.

Figure 18 shows the average saliency map for the high-risk class. An initial inspection suggests that the generated saliency map is working as expected, as the frames near where lifts tend to occur (the 100-200 range) tend to be marked as important relative to other frames. On a per-sensor basis, the upper arm and back sensors have higher occurrences of important frames relative to other sensors, which suggests that these sensors are the most important sensors when

determining lift risk. Other sensors with notable “hotspots” are the right wrist and waist sensors, although there are significantly fewer of these. The saliency maps for low- and medium-risk lifts showed similar patterns.

Therefore, good picks for sensors to keep include the back and upper arm sensors, followed by wrist and waist. This makes intuitive sense, as the back sensor (and possibly the waist sensor) can easily pick up lumbar motion common when lifting objects, while the upper arm and wrist sensors can track arm motion used to lift.

5.3.2 Sensor Results

To verify the saliency mapping results and determine how well the system would function with fewer sensors, the classification model was trained and evaluated using different subsets of the available sensors in the training dataset. The saliency map results show that the most influential sensors with regard to risk classification were upper arm and back, followed by either the right wrist or the waist. Different training subsets were chosen to analyze how the identified sensors affect the capabilities of the model when trained.

The first subset included all four of the identified important sensors. Another included just the upper arm and back, since these sensors were most clearly defined in the saliency maps. Two more subsets included the two most important sensors, plus either the right wrist or the waist, to determine which of the following two are more important. The left wrist sensor was added to the main four in another subset, since it should have similar information to the right wrist sensor during a lift action. Finally, some subsets were constructed using the sensors identified as less important. Table VII summarizes the subsets used and their results.

Table VII: Balanced Accuracy of Sensor Subsets

Included Sensors	Balanced Accuracy
All	0.886
UArm, Back, RtWt, Waist	0.828
UArm, Back, RtWt	0.825
UArm, Back, LtWt	0.816
UArm, Back, RtWt, LtWt	0.814
UArm, Back, Waist	0.742
UArm, Back	0.658
LtWt, Waist, Thigh	0.776
Waist, Thigh	0.401

Including only the upper arm and back sensors results in a significant reduction in balanced accuracy for lift classification. Adding the waist sensor resulted in a significant increase in accuracy, but adding the right wrist resulted in a much larger increase. Adding both only resulted in a small increase over just the right wrist. Adding the left wrist had negligible impact on the subset with the right wrist. Using just the waist and thigh sensors resulted in very poor performance, but adding the left wrist boosted this considerably, almost back up to levels with the four saliency-identified sensors.

Every subset with upper arm and back sensors achieved >60% balanced accuracy, suggesting that these sensors are important. Adding the right wrist sensor resulted in a much larger performance boost than adding the waist sensor, suggesting that the waist sensor is not as important

as the right wrist. Surprisingly, including the upper arm, back, and right wrist sensors (the “important" sensors) resulted in close to the same performance as including the left wrist, waist, and thigh sensors (the less important sensors). When the left wrist sensor was removed from the latter set, balanced accuracy fell considerably. However, including the left wrist with the important sensors did not significantly change the results. This suggests that utilizing both wrists is redundant, but having at least one is important. This makes intuitive sense since both wrists should move in sync to lift an object and therefore have similar information. It is possible that the right wrist is preferred when both are included since most of the population is right-handed, and this was reflected in the saliency map. These results generally corroborate the saliency map results, with upper arm and back sensors providing the bulk of predictive capabilities, with additional boosts provided by either one of the wrists, waist, and thigh sensor, in decreasing order of usefulness.

Utilizing all sensors resulted in the best performance, rather than any of the subsets. It is important to note that the performance on the subsets was tested using the same hyperparameters and architecture as the full set, and which were tuned to input data with all six sensors. There is a higher chance of overfitting and other common training problems as the amount of data decreases and the model’s parameters begin to significantly outnumber the discriminative features of the dataset. Therefore, the accuracy scores shown are not representative of the (correctly tuned) model's performance on a smaller number of sensors and should only be considered a point of comparison between sensors. Tuning the model with a smaller number of sensors may reduce the drop in performance from the lost sensors.

Chapter 6. Conclusions and Future Work

6.1. Conclusions

Machine learning, particularly LSTM networks, are often used in the context of HAR problems for detection and classification of human actions. Their success has enabled rapid growth in both academic and industrial settings, providing opportunities for improvement of worker safety and health and quality of life improvements across the general population. However, the extensive research in this field primarily focuses common household activities that can be easily detected by even an untrained eye, such as walking, running, and jumping. This severely limits the capabilities of these models in specialized settings, especially when data may be limited and difficult to separate. The NIOSH lifting dataset is an example of one such dataset, where each class of activity is a slight modification to a standard action. It takes an expert observer to determine which class is occurring, meaning the classification setting is much more difficult and commonly used solutions may fail.

This thesis presents a machine learning system for detection and classification of different types of lifting action at varying levels of risk for the worker. Both models achieve high levels of performance on a dataset significantly smaller than standard publicly available datasets for recognition on IMU data. The use of residual blocks for feature extraction reduces the need for expert processing of the data and allows for deeper model architectures, while LSTM layers capture temporal relations of the features to assist in the final decision. Reduced need for preprocessing of data improves the capabilities of implementations on edge devices, where the compute capabilities required for common preprocessing steps may not be available. This system allows for accurate monitoring in a difficult setting and may extend to other specialized tasks as well.

6.2. Future Work

A primary avenue for future work is researching the applicability of the models' architectures for other specialized tasks. The current configuration performs well on the NIOSH dataset, but it is unclear if the same will hold in a different setting without significant modification to architecture or parameters. In a setting with more data, increasing the number of layers in the network may improve performance further, supported by the skip connections in the residual blocks. Additionally, there may be improvements gained in this setting with additional adjustments to the proposed model, perhaps with layer normalization or other standardization techniques.

The application of further transformations for data augmentations and their combinations poses another interesting topic for future exploration. In this work, a single augmentation resulted in a significant increase in model performance for the classification task. However, the set of transformations applied was limited to determine applicability. Utilizing more sophisticated transformations as well as different combinations may result in further gains.

Thus far, the system has been trained and evaluated on laboratory data collected and stored on disk. The next logical step is to test the system on real-time, real-life data collected on a factory floor or workshop. Assessing the system's capabilities for real-time detection and classification on data as it will appear during regular work operations is a crucial step towards full-scale implementation, and work has already begun in this direction. Additionally, reducing the sensor requirements would increase the system's utility and ability to be deployed in work settings.

Bibliography

[1] "Work Practices Guide for Manual Lifting.," September 28, 2020. https://doi.org/10.26616/NIOSHPUB81122.

[2] National Research Council (US) and Institute of Medicine (US) Panel on Musculoskeletal Disorders and the Workplace. *Musculoskeletal Disorders and the Workplace: Low Back and Upper Extremities*. Washington (DC): National Academies Press (US), 2001. http://www.ncbi.nlm.nih.gov/books/NBK222440/.

[3] "Musculoskeletal Disorders and Workplace Factors. A Critical Review of Epidemiologic Evidence for Work-Related Musculoskeletal Disorders of the Neck, Upper Extremity, and Low Back.," October 6, 2020. https://doi.org/10.26616/NIOSHPUB97141.

[4] "Low Back Pain Fact Sheet | National Institute of Neurological Disorders and Stroke." Accessed September 25, 2021. https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Low-Back-Pain-Fact-Sheet.

[5] "Applications Manual for the Revised NIOSH Lifting Equation.," October 2, 2020. https://doi.org/10.26616/NIOSHPUB94110.

[6] Splittstoesser, Riley, Daniel Edward O'Farrell, John Hill, Terrence McMahon, Nikhil Sastry, and Mark Tiemeier. "2005 ACGIH Lifting TLV: Employee-Friendly Presentation and Guidance for Professional Judgment," May 22, 2017. https://doi.org/10.2172/1358182.

[7] Barim, Menekse S., Ming-Lun Lu, Shuo Feng, Grant Hughes, Marie Hayden, and Dwight Werren. "Accuracy of An Algorithm Using Motion Data Of Five Wearable IMU Sensors For Estimating Lifting Duration And Lifting Risk Factors." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 63, no. 1 (November 2019): 1105–11. https://doi.org/10.1177/1071181319631367.

[8] Hochreiter, Sepp. "Recurrent Neural Net Learning and Vanishing Gradient." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 1998.

[9] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long Short-Term Memory." *Neural Computation* 9, no. 8 (November 1, 1997): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

[10] Sundermeyer, Martin, Ralf Schluter, and Hermann Ney. "LSTM Neural Networks for Language Modeling," n.d., 4.

[11] Zhou, Chunting, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. "A C-LSTM Neural Network for Text Classification." *ArXiv:1511.08630 [Cs]*, November 30, 2015. http://arxiv.org/abs/1511.08630.

[12] Zhao, Zheng, Weihai Chen, Xingming Wu, Peter C. Y. Chen, and Jingmeng Liu. "LSTM Network: A Deep Learning Approach for Short-term Traffic Forecast." *IET Intelligent Transport Systems* 11, no. 2 (March 2017): 68–75. https://doi.org/10.1049/iet-its.2016.0208.

[13] Gers, Felix A, Nicol N Schraudolph, and Jurgen Schmidhuber. "Learning Precise Timing with LSTM Recurrent Networks," n.d., 29.

[14] Shi, Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting." *ArXiv:1506.04214 [Cs]*, September 19, 2015. http://arxiv.org/abs/1506.04214.

[15] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning." *Nature* 521, no. 7553 (May 2015): 436–44. https://doi.org/10.1038/nature14539.

[16] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *ArXiv:1512.03385 [Cs]*, December 10, 2015. http://arxiv.org/abs/1512.03385.

[17] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity Mappings in Deep Residual Networks." *ArXiv:1603.05027 [Cs]*, July 25, 2016. http://arxiv.org/abs/1603.05027.

[18] Weiss, Gary M., Jeffrey W. Lockhart, Tony T. Pulickal, Paul T. McHugh, Isaac H. Ronan, and Jessica L. Timko. "Actitracker: A Smartphone-Based Activity Recognition System for Improving Health and Well-Being." In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 682–88, 2016. https://doi.org/10.1109/DSAA.2016.89.

[19] Chen, Yuqing, and Yang Xue. "A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer." In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 1488–92, 2015. https://doi.org/10.1109/SMC.2015.263.

[20] Snyder, Kristian, Brennan Thomas, Ming-Lun Lu, Rashmi Jha, Menekse S. Barim, Marie Hayden, and Dwight Werren. "A Deep Learning Approach for Lower Back-Pain Risk Prediction during Manual Lifting." *PLOS ONE* 16, no. 2 (February 19, 2021): e0247162. https://doi.org/10.1371/journal.pone.0247162.

[21] Cook, Diane, Kyle Feuz, and Narayanan Krishnan. "Transfer Learning for Activity Recognition: A Survey." *Knowledge and Information Systems* 36 (September 1, 2013): 537–56. https://doi.org/10.1007/s10115-013-0665-3.

[22] Rodríguez-Moreno, Itsaso, José María Martínez-Otzeta, Basilio Sierra, Igor Rodriguez, and Ekaitz Jauregi. "Video Activity Recognition: State-of-the-Art." *Sensors* 19, no. 14 (January 2019): 3160. https://doi.org/10.3390/s19143160.

[23] Ignatov, Andrey. "Real-Time Human Activity Recognition from Accelerometer Data Using Convolutional Neural Networks." *Applied Soft Computing* 62 (January 1, 2018): 915–22. https://doi.org/10.1016/j.asoc.2017.09.027.

[24] Anguita, Davide, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. "A Public Domain Dataset for Human Activity Recognition Using Smartphones." *Computational Intelligence*, 2013, 6.

[25] Ordóñez, Francisco Javier, and Daniel Roggen. "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition." *Sensors* 16, no. 1 (January 2016): 115. https://doi.org/10.3390/s16010115.

[26] Roggen, Daniel, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, et al. "Collecting Complex Activity Datasets in Highly Rich Networked Sensor Environments." In *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, 233–40, 2010. https://doi.org/10.1109/INSS.2010.5573462.

[27] Zappi, Piero, Daniel Roggen, Elisabetta Farella, Gerhard Tröster, and Luca Benini. "Network-Level Power-Performance Trade-Off in Wearable Activity Recognition: A Dynamic Sensor Selection Approach." *ACM Transactions on Embedded Computing Systems* 11, no. 3 (September 1, 2012): 68:1-68:30. https://doi.org/10.1145/2345770.2345781.

[28] Hammerla, Nils Y., Shane Halloran, and Thomas Ploetz. "Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables." *ArXiv:1604.08880 [Cs, Stat]*, April 29, 2016. http://arxiv.org/abs/1604.08880.

[29] Ravi, Daniele, Charence Wong, Benny Lo, and Guang-Zhong Yang. "Deep Learning for Human Activity Recognition: A Resource Efficient Implementation on Low-Power Devices." In *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 71–76, 2016. https://doi.org/10.1109/BSN.2016.7516235.

[30] Bhattacharya, Sourav, and Nicholas D. Lane. "From Smart to Deep: Robust Activity Recognition on Smartwatches Using Deep Learning." In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 1–6, 2016. https://doi.org/10.1109/PERCOMW.2016.7457169.

[31] Guan, Yu, and Thomas Ploetz. "Ensembles of Deep LSTM Learners for Activity Recognition Using Wearables." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, no. 2 (June 30, 2017): 1–28. https://doi.org/10.1145/3090076.

[32] Cavallo, Andrea, Atesh Koul, Caterina Ansuini, Francesca Capozzi, and Cristina Becchio. "Decoding Intentions from Movement Kinematics." *Scientific Reports* 6, no. 1 (November 15, 2016): 37036. https://doi.org/10.1038/srep37036.

[33] Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." *ArXiv:1412.6980 [Cs]*, January 29, 2017. http://arxiv.org/abs/1412.6980.

[34] Hawkins, Douglas M. "The Problem of Overfitting." *Journal of Chemical Information and Computer Sciences* 44, no. 1 (January 1, 2004): 1–12. https://doi.org/10.1021/ci0342472.

[35] Prechelt, Lutz. "Early Stopping - But When?" In *Neural Networks: Tricks of the Trade*, edited by Genevieve B. Orr and Klaus-Robert Müller, 55–69. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1998. https://doi.org/10.1007/3-540-49430-8_3.

[36] Arlot, Sylvain, and Alain Celisse. "A Survey of Cross-Validation Procedures for Model Selection." *Statistics Surveys* 4, no. none (January 2010): 40–79. https://doi.org/10.1214/09-SS054.

[37] Saeb, Sohrab, Luca Lonini, Arun Jayaraman, David C. Mohr, and Konrad P. Kording. "The Need to Approximate the Use-Case in Clinical Machine Learning." *GigaScience* 6, no. 5 (May 1, 2017). https://doi.org/10.1093/gigascience/gix019.

[38] Probst, Philipp, Anne-Laure Boulesteix, and Bernd Bischl. "Tunability: Importance of Hyperparameters of Machine Learning Algorithms." *Journal of Machine Learning Research* 20, no. 53 (2019): 1–32.

[39] Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. "Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors." *ArXiv:1207.0580 [Cs]*, July 3, 2012. http://arxiv.org/abs/1207.0580.

[40] Cortes, Corinna, Mehryar Mohri, and Afshin Rostamizadeh. "L2 Regularization for Learning Kernels." *ArXiv:1205.2653 [Cs, Stat]*, May 9, 2012. http://arxiv.org/abs/1205.2653.

[41] Mosley, Lawrence. "A Balanced Approach to the Multi-Class Imbalance Problem." *Graduate Theses and Dissertations*, January 1, 2013. https://doi.org/10.31274/etd-180810-3375.

[42] Kotsiantis, Sotiris, Dimitris Kanellopoulos, and Panayiotis Pintelas. "Handling Imbalanced Datasets: A Review," 2006, 12.

[43] Shorten, Connor, and Taghi M. Khoshgoftaar. "A Survey on Image Data Augmentation for Deep Learning." *Journal of Big Data* 6, no. 1 (July 6, 2019): 60. https://doi.org/10.1186/s40537-019-0197-0.

[44] Dyk, David A van, and Xiao-Li Meng. "The Art of Data Augmentation." *Journal of Computational and Graphical Statistics* 10, no. 1 (March 1, 2001): 1–50. https://doi.org/10.1198/10618600152418584.

[45] Kim, Mooseop, and Chi Yoon Jeong. "Label-Preserving Data Augmentation for Mobile Sensor Data." *Multidimensional Systems and Signal Processing* 32, no. 1 (January 1, 2021): 115–29. https://doi.org/10.1007/s11045-020-00731-2.

[46] Um, Terry Taewoong, Franz Michael Josef Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. "Data Augmentation of Wearable Sensor Data for Parkinson's Disease Monitoring Using Convolutional Neural Networks." *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, November 3, 2017, 216–20. https://doi.org/10.1145/3136755.3136817.

[47] Niswander, Wesley, Wei Wang, and Kimberly Kontson. “Optimization of IMU Sensor Placement for the Measurement of Lower Limb Joint Kinematics.” *Sensors* 20, no. 21 (January 2020): 5993. https://doi.org/10.3390/s20215993.

[48] Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” *ArXiv:1312.6034 [Cs]*, April 19, 2014. http://arxiv.org/abs/1312.6034.