Engineering Applications of Artificial Intelligence 126 (2023) 106992

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Material handling machine activity recognition by context ensemble with gated recurrent units

Kunru Chen [a,*], Thorsteinn Rögnvaldsson [a], Sławomir Nowaczyk [a], Sepideh Pashami [a], Jonas Klang [b], Gustav Sternelöv [b]

[a] Center for Applied Intelligent Systems, Halmstad University, Kristian IV:s vag 3, Halmstad, 301 18, Halland, Sweden
[b] Toyota Material Handling Manufacturing Sweden AB, Svarvargatan 8, Mjolby, 595 35, Ostergotland, Sweden

ARTICLE INFO

Keywords:
Context ensemble
Machine activity recognition
Gated recurrent unit
Material handling
Productivity monitoring

ABSTRACT

Research on machine activity recognition (MAR) is drawing more attention because MAR can provide productivity monitoring for efficiency optimization, better maintenance scheduling, product design improvement, and potential material savings. A particular challenge of MAR for human-operated machines is the overlap when transiting from one activity to another: during transitions, operators often perform two activities simultaneously, e.g., lifting the fork already while approaching a rack, so the exact time when one activity ends and another begins is uncertain. Machine learning models are often uncertain during such activity transitions, and we propose a novel ensemble-based method adapted to fuzzy transitions in a forklift MAR problem. Unlike traditional ensembles, where models in the ensemble are trained on different subsets of data, or with costs that force them to be diverse in their responses, our approach is to train a single model that predicts several activity labels, each under a different context. These individual predictions are not made by independent networks but are made using a structure that allows for sharing important features, i.e., a context ensemble. The results show that the gated recurrent unit network can provide medium or strong confident context ensembles for 95% of the cases in the test set, and the final forklift MAR result achieves accuracies of 97% for driving and 90% for load-handling activities. This study is the first to highlight the overlapping activity issue in MAR problems and to demonstrate that the recognition results can be significantly improved by designing a machine learning framework that addresses this issue.

1. Introduction

Machine activity recognition (MAR) is a research area dedicated to developing data-driven methods for monitoring machines. It offers various benefits, including improved productivity monitoring, enhanced maintenance scheduling, better product design, and potential cost savings. While the majority of MAR research has focused on construction equipment, it is important to also study material handling machines. Material handling machines play a vital role in the manufacturing industry and are utilized in almost every warehouse or manufacturing site globally. Recognizing the potential advantages of productivity monitoring in the material handling sector, this paper delves into MAR specifically for one of its key components — the forklift.

Generally, the forklift MAR problem is challenging since there is quite a variation in how drivers perform activities. Loads and tasks are two main elements that can greatly decide forklift operations. Delivery of very heavy loads (which require extra care), very light loads (that cannot be detected by weight sensors), loads at high heights, or loads with irregular shapes, e.g., cages and pipes, require drivers to adapt their behaviors. In some cases, it requires them to be more careful with the load and more aware of the operational environment; in others, the circumstances may require high speed at the cost of precision. Furthermore, drivers develop their own preferred practice on how to do operations. Therefore, activity patterns differ from the by-the-book usage defined by domain experts.

Additionally, apart from typical delivery (picking up a load at one place, transporting it, and placing it somewhere else), other tasks can also be performed. Examples of different tasks are goods organization with a sort-out, where a series of forklift movements happens consecutively (without any driving), and floor-level load handling, where the fork is used to push the load (instead of lifting or lowering). Therefore, many challenges remain before MAR becomes a mainstream tool in industrial forklift operations.

This paper focuses on a particular issue in forklift MAR: stable recognition during the fuzzy transitions between machine activities.

* Corresponding author.
E-mail address: kunru.chen@hh.se (K. Chen).

https://doi.org/10.1016/j.engappai.2023.106992
Received 8 August 2022; Received in revised form 24 July 2023; Accepted 12 August 2023
Available online 28 August 2023

The boundary between adjacent activities is not always clear-cut, as forklift drivers often perform two activities simultaneously. Hence, forklift operations can be separated into *overlapping* and *non-overlapping* activities; and recognizing forklift activities accurately becomes more challenging when a significant portion of time is dedicated to *overlapping* activity. Typically, the longer a transition takes, the smoother it is, yet the more difficult it is to recognize correctly. Intuitively, this fuzzy transition issue probably occurs also in activities with other types of equipment, e.g. construction equipment. However, to the best of our knowledge, no previous paper on MAR has mentioned this issue.

Reliable handling of the fuzzy activity transitions is very important if the context and order of operations are needed for accurate MAR, e.g. when activities are expected to be performed in a certain order, an order that can be used to improve the MAR. This is the case for forklifts, where, e.g., the presence or absence of a load on the forks can be inferred from the activity context even if the load is too light to be detected by the on-board load sensor.

To handle the fuzzy transitions, we propose to structure machine learning models as context ensembles, i.e., that the predictions obtained from different contexts form an ensemble. Here, the different contexts are "looking back to the past" and "looking ahead to the future", both to varying degrees. The classification of a segment of data may be different when looking at past segments or at future segments. Being able to peer into the future has been shown to make a significant difference compared to using past data in human activity recognition (Ali Hamad et al., 2020). Two advantages are brought by context ensembles, compared with basic ensemble methods that use subsets of data (or different model parameters) to train several individual diverse models. First, only one model needs to be trained in the proposed method because predictions in the ensemble are collected from different contexts rather than from different models. In an industrial setting, this can provide significant time savings. Second, the model is trained to predict activity labels with different contexts, and it functions as a regularization to prevent overfitting during training.

In this paper, gated recurrent units (GRU) are trained to recognize routine forklift activities with shifted time windows, i.e., different contexts. The results show that the context ensemble can significantly improve the stability of predictions during activity transitions; after the post-processing, the proposed method obtains the best scores in all driving and load-handling activities. Overall, this paper presents our work on a real-world engineering problem and two novel aspects: (1) it suggests the utilization of gated recurrent units (GRU) for MAR problems, and (2) it suggests using shifted time windows to handle overlapping activities, a topic that has not been covered in any previous work on machine learning for MAR.

2. Background

Sherafat et al. (2020) and Harichandran et al. (2021) present two recent state-of-the-art reviews of the MAR field. Sherafat et al. (2020) group MAR methods according to the data used: "kinematic-based methods" with data collected from accelerometers and/or gyroscopes (Ahn et al., 2015; Akhavian and Behzadan, 2015; Kim et al., 2018; Rashid and Louis, 2020b; Slaton et al., 2020), "vision-based methods" with images and videos from cameras (Gong and Caldas, 2009; Golparvar-Fard et al., 2013; Mahami et al., 2019; Kim and Chi, 2020; Wu et al., 2021; Jung et al., 2021), and "audio-based methods" with sounds from microphones (Cheng et al., 2017; Rashid and Louis, 2020a). The two reviews show that essentially all MAR approaches so far base the activity recognition on data collected with sensors that are external to the machine, very few use internal sensor data. Three examples with internal sensors are the works by Shi et al. (2020), Fischer et al. (2021a), and Fischer et al. (2021b). Using external sensors can be problematic; they need separate calibration, they need to be properly (and often carefully) placed, and they can be negatively affected by external (e.g., weather) conditions. Furthermore, awareness of external sensors can potentially influence operator behavior so that it does not reflect normal operation. To avoid this, signals from the on-board Controller Area Network (CAN) are used in this paper. CAN is an industrial standard protocol for internal communication on a vehicle, which means that CAN data is streaming and multivariate. Furthermore, since signals on the CAN are used for vehicle control and monitoring, they are carefully tested during product development and their quality is good and stable.

Most existing MAR approaches share a common logic: first, to develop a machine learning model to recognize pieces of data, i.e., using small sliding windows to segment the data, and then to post-process the raw recognition result with rules (Shi et al., 2020) or filters (Jakobsson et al., 2020). The motivation for the post-processing step is that activities have a natural order, so past or future activities should be meaningful when inferring the ongoing activity. However, there is potential to improve the first step, i.e., before post-processing, such as by building a more advanced machine learning model (Slaton et al., 2020; Akinosho et al., 2020), to achieve more accurate and stable recognition directly. Some MAR researchers (Fischer et al., 2021a; Rashid and Louis, 2020b; Jakobsson et al., 2020) have proposed recurrent neural networks (RNN) and convolutional neural networks (CNN) combined with relatively large sliding window sizes. Similarly, in this paper, we use recurrent models to capture the concordance between forklift activities.

The "working cycle" concept has become popular recently as useful for MAR and productivity monitoring (Chen et al., 2020; Kim and Chi, 2019; Shi et al., 2020; Wu et al., 2021). A working cycle is the series of activities needed to finish a task. One example (Shi et al., 2020) is the working cycle for digging: including *1st pre-digging*, *2nd pre-digging*, *digging*, *1st lifting*, *2nd lifting*, *unloading*, and *swinging*. Wu et al. (2021) used the working cycle duration to filter out abnormal operations from the data. They used a condition of working cycle duration of shorter than six seconds to identify abnormal operations, which reveals that excavators (at least in their data set) have a fairly stable working scenario: performing cyclical and repetitive operations without changing the position of the equipment. This is quite different from how forklifts are used.

3. Data description

3.1. Data collection and characteristics

This paper uses CAN signals logged on forklifts with a compact CAN logger, the Vector GL 1000. Two data sets are involved in this paper: a labeled data set collected from a Toyota development lab in Sweden and an unlabeled data set collected from a Norwegian warehouse during normal operation. In both cases, 262 CAN signals are sampled from eight units in the CAN bus, with a sample rate of 10 Hz. The CAN signals are, e.g., forklift wheel speed, wheel direction, height of forks, reach position for forks, speed of lift motor, estimated load weight, steering angle command, lifting/lowering command, reach command, and suchlike. In order to label the data, a camera was mounted in the cabin, capturing a view with both the control panel and the forks. There is a timestamp in each CAN entry, which can be used to synchronize with the time information provided by the camera. Two subsets, one with 58 min data and one with 27 min data, are included in the labeled data set. The longer 58 min subset corresponds to operations performed under the constraint of the lab environment, and it is used as training data. The shorter 27 min one is used as test data where flexible forklift usage is presented. The first half of the short data set are frequent and regular by-the-book operations, similar to the 58 min training set; the second half of the short data set presents more the maneuvering of the forklift rather than the load-handling of it, and there is an occasion when the forklift was driving with a load that is too light to be detected by the forklift weight sensor.

There is no ground truth available for the unlabeled data set, so it is unclear how far it is from the by-the-book cases. However, since it is collected during normal operation, it is expected to contain more flexible operations than the labeled data.

Table 1
Statistics for the activity labels and the activity duration in the 58 min data set. The label of "Other" means any activity that belongs to neither *driving* nor *load-handling* activity. This can be, e.g., waiting for traffic to pass, picking up orders, checking that the unloading was done correctly, or chatting with a colleague.

Activity	Domain expert label				Person 1	Person 2
	# Cases	Average (s)	Min. (s)	Max. (s)	Label agreement	with domain expert
Other	18	44.5	12	162	82.1%	87.4%
Drive without load	21	38.5	5	151	87.5%	87.9%
Drive with load	22	23.6	2	98	91.3%	89.9%
Take Load	23	21.2	2	55	87.6%	87.4%
Leave Load	22	14.5	6	25	93.4%	94.3%

3.2. Activity duration and transition

A domain expert labeled the data from the Toyota lab using the videos recorded during data collection, and an activity label is assigned to every second of the data. Table 1 shows the target activities and the statistics of activity duration in the 58 min data set, where 22 working cycles in total are performed. One can see that the duration of an activity can vary a lot between two activities, e.g., *Drive with Load* and *Leave Load*, or inside a single activity, e.g., the minimum and the maximum length of *Drive without Load*. Moreover, the generally short duration for the *Leave Load* activity causes a noticeable imbalance in the class distribution.

To some degree, activities overlap in the transitions between them. As the initial activity gradually approaches its completion, the subsequent activity progressively takes over, making the recognition during the transition period particularly challenging. Since the time spent on every transition is different and unpredictable, getting precise statistics of *non-overlapping* and *overlapping* periods is impractical. However, it can be approximated by the disagreement between people who label the same data. Table 1 shows activity labeling results from two non-domain experts, also using the video data (more explanation on the metrics can be found in Section 4.5). Unsurprisingly, they do not completely agree with the expert label; even watching the same video for doing the labeling, human annotators make different judgment calls. In particular, the disagreement about *Other*, *Drive without Load*, and *Take Load* are higher (than the rest two activities), translating longer time spent on activity transitions.

4. Method

In this section, we explain the two key elements in the proposed method; the GRU and the context ensemble. The method (detailed in Appendix A) consists of four steps: generating pseudo-labeled data (Lee, 2013), training GRU networks (Cho et al., 2014), forming context ensembles and applying post-processing (Chen et al., 2022). let $\mathcal{Y}$ denote the output space of a K-way classification problem where $\mathcal{Y}$ is a finite set, i.e., $\mathcal{Y} = \{1, 2, \dots, K\}$. Consider a time-series $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t, \dots, \mathbf{x}_n\}$ in the input space $\mathcal{X}$, and $\mathbf{x}_t$ is the vector of input signals at time t with an activity label $y_t \in \mathcal{Y}$. Sliding windows with a fixed window size m are applied, and every sliding window contains several input vectors, e.g., the sliding window w_t consist of $\{\mathbf{x}_{t-m+1}, \mathbf{x}_{t-m+2}, \dots, \mathbf{x}_t\}$. The stride of the sliding window is one time step, which means the window moves forward (along the time axis) by one time step each time.

4.1. Baseline classifier

The baseline classifier is trained with only the labeled data (the 58 min data set), illustrating to what degree the problem can be solved by merely applying a supervised learning technique. The task is to take the sliding windows as input and return the last label in that window. For example, when considering the input window w_t, the baseline classifier should learn to recognize the last activity in that window, i.e., y_t. Random forest (RF) is chosen as the baseline classifier because it achieved the best classification performance among algorithms (including logistic regression and support vector machine) in previous work, which also utilized CAN signals but for excavator MAR (Shi et al., 2020). The baseline classifier is denoted baseline-RF.

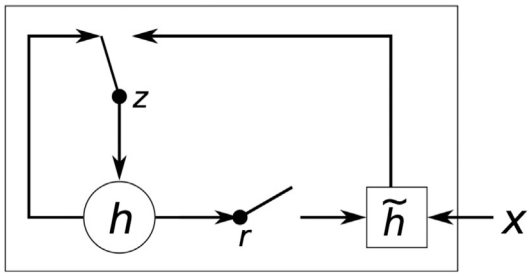

Fig. 1. An illustration of a GRU neuron (Cho et al., 2014).

According to the pseudo-label method proposed by Lee (2013), only those predictions with high model confidence should be selected as pseudo-labels. Following this procedure, we applied the baseline-RF to the unlabeled data set, and pseudo-labels with high prediction probabilities were selected for training the GRU network. Although there is no guarantee that all pseudo-labels are correct and informative, they represent the knowledge learned by the baseline-RF and have been shown to improve MAR results (Chen et al., 2022).

4.2. The Gated Recurrent Unit (GRU)

Introduced by Cho et al. (2014), the GRU network is a popular member of the RNN family. The idea of the simple RNN (Elman, 1990) is to decide outputs based on the inputs and the hidden state, which is calculated from previous hidden states or previous outputs. As for the GRU, it provides advanced control of the information in the hidden state with an update gate and a reset gate. Generally, it can decide what information from the past can be removed from the hidden state (as it is irrelevant to the current state) and what information from the current inputs should be added (as it can be important to the future). The GRU has fewer parameters than the long short-term memory (Hochreiter and Schmidhuber, 1997), where a unit consists of three gates and a cell structure. The use of GRUs for MAR is inspired by the fact that GRUs have been used with some success in studies on human activity recognition, e.g., Kolkar et al. (2022), Mohsen (2023), Shiri et al. (2023), and Sun et al. (2022).

Fig. 1 shows the structure of a single GRU h, where r is the *reset* gate and z is the *update* gate. A GRU network has several such units, denoted h_j (with corresponding r_j and z_j), which are updated with the equations below:

$$r_j = \sigma\left(\left[\mathbf{V}_r\mathbf{x}\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}\right]_j\right),$$

$$z_j = \sigma\left(\left[\mathbf{V}_z\mathbf{x}\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right).$$

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + \left(1 - z_j\right)\tilde{h}_j^{\langle t\rangle},$$

$$\tilde{h}_j^{\langle t\rangle} = \phi\left(\left[\mathbf{V}\mathbf{x}\right]_j + \left[\mathbf{U}\left(\mathbf{r}\odot\mathbf{h}_{\langle t-1\rangle}\right)\right]_j\right).$$

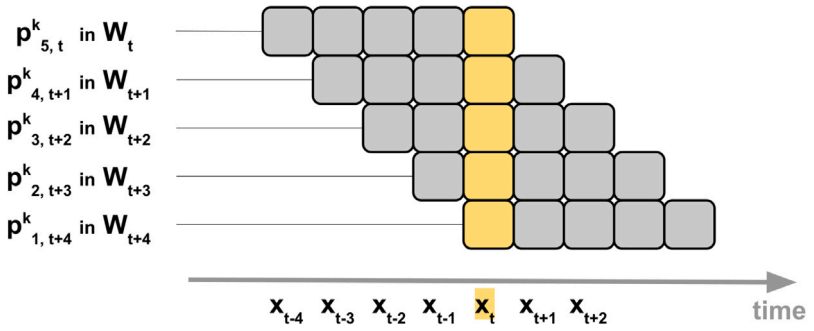

Fig. 2. An illustration of the meaning of the same input $\mathbf{x}_t$ in different windows from w_t to w_{t+4}.

The input at time t is the vector $\mathbf{x}^{(t)}$. The letters $\mathbf{V}$ and $\mathbf{U}$ denote parameter matrices (the bias parameter is included in $\mathbf{V}$). The boldface notation $\mathbf{h}$ and $\mathbf{r}$ denote the vectors with all the values h_j and r_j, and $[\cdot]_j$ indicates taking the jth element of a vector. Originally, σ is a logistic function and ϕ a hyperbolic tangent function. At initialization, $h_j^{(0)} = 0$ for all j.

The GRU networks are trained to predict multiple labels for every input, supporting the context ensemble idea (further explained in Section 4.3). For the input window w_t, the GRU networks are trained to predict all target labels $\{y_{t-m+1}, y_{t-m+2}, \ldots, y_t\}$, i.e the label for each time step input in w_t, from $\mathbf{x}_{t-m+1}$ to $\mathbf{x}_t$. Softmax is used as the activation function in the output layer of the network. Thus, for each timestep within the window, the network should return the prediction probabilities for each of the five target activities. In our notation, we let $p_{i,t}^k$ denote the network output of input window w_t, where $i \in \{1, 2, \ldots, m\}$ is the index of the predicted time step and $k \in \{1, 2, 3, 4, 5\}$ is the index of the softmax component representing the particular activity. For example, the output $p_{4,t}^5$ indicates the prediction probability for the fifth activity at the fourth time step in the input window w_t.

4.3. Context ensemble

A committee is often a better choice than a single expert, and the committee also provides an estimate of the variation in opinions. The context ensemble is based on the idea that a model can have different opinions about the activity in a particular time step depending on if it is looking into the past or the future, i.e., if the input vector is located early in the sliding window, or late. In theory, this could be achieved with one long input window, but we choose to use a committee of partially overlapping time windows instead. Fig. 2 illustrates the setup with an example window size $m = 5$: input $\mathbf{x}_t$ is the last time step of window w_t, and therefore models need to predict what happened at time t based on the past data points; in the window w_{t+4}, on the other hand, input $\mathbf{x}_t$ must be used by the model to predict what will happen in the future.

The context ensemble $\mathbf{C}_t = \{p_{i,t+m-i}^k\}_{i=1}^m$ is formed by collecting all outputs for the same time step vector $\mathbf{x}_t$. Once the activity index k is expanded, $\mathbf{C}_t$ is an $m \times 5$ matrix. Fig. 3 shows $\mathbf{C}_t$ for the example above with a window size of $m = 5$. Here, $\mathbf{C}_t$ consists of five softmax results from five different windows, showing all the (potentially different) prediction probabilities of activities at time t.

There are many possible approaches to utilizing the context ensembles; perhaps the most straightforward way is to simply use "majority vote" over the rows in $\mathbf{C}_t$. However, this discards much information from other classes (other rows). To use more information in the context ensemble, we calculate the average across the time window:

$$s_t^k = \frac{1}{m} \sum_{i=1}^{m} p_{i,t+m-i}^k, \tag{1}$$

and the final predicted activity $\hat{y}_t$ is the index of the maximum s_t^k among the k activities, defined as:

$$\hat{y}_t = \underset{k}{\text{argmax}}(s_t^k). \tag{2}$$

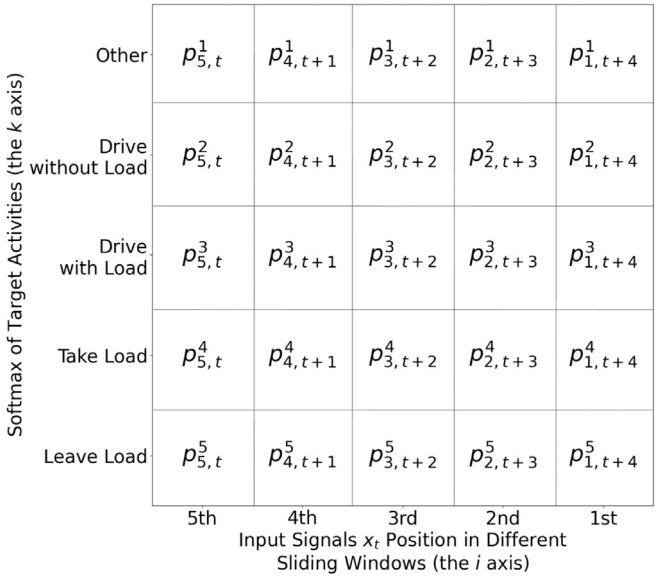

Fig. 3. An example context ensemble $\mathbf{C}_t$ matrix, when the window size is five time steps, and the number of activities is also five.

Additionally, to analyze the confidence of the context ensemble, we define strong, medium, and weak ensemble based on the maximum element in the average vector, i.e., $max(s_t^k)$. Context ensembles where $\max\left(s_t^k\right) > 0.95$ are denoted *strong*, i.e. they are very confident. Context ensembles where $\max\left(s_t^k\right) < 0.5$ are denoted *weak*, i.e. they are weakly confident. Context ensembles where $\max\left(s_t^k\right) \in [0.5, 0.95]$ are denoted *medium*, i.e. they are medium confident.

4.4. Post-processing

The post-processing builds on the expected order of activities and corrects erroneous transitions (Chen et al., 2022). Those forklift activities are supposed to happen in a particular order: *Drive without Load* → *Take Load* → *Drive with Load* → *Leave Load*, followed by a new working cycle starting from the first activity, i.e., *Drive without Load* again. A typical mistake from the models can be changing to a wrong load-handling after a driving, e.g., predictions wrongly switched from *Drive with Load* to *Take Load* due to the difficulties of recognizing *Leave Load*. In this case, post-processing that considers relations between two consecutive activities can improve the recognition results.

The rule-based post-processing works differently for wrong transitions occurring in two cases. For transitions between driving and load-handling activities, the post-processing rule forces the second activity in the transition to follow the logic correctly, assuming that the preceding activity is correct. For example, if an illegal transition from

Table 2
Networks structures.

Model code	Model structure
MLP	Input → Dense(256)[1] → Dense(128) → Dense(32) → Linear(3) → Dense(10) → Dense(k)[1] → Softmax as the output
MLP-CE	Input → Dense(256) → Dense(128) → Dense(32) → Linear(3) → Dense(10) → Linear($m \times k$)[2] → Reshape(m, k) → Softmax as the output
GRU-CE	Input → GRU(128) → GRU(32) → Dense(16) → Linear(3) → Dense(10) → Linear($m \times k$) → Reshape(m, k) → Softmax as the output

1 Dense is a fully connected linear layer with ReLu activation.
2 There are $k = 5$ target activities.
3 Window size is $m = 5$.

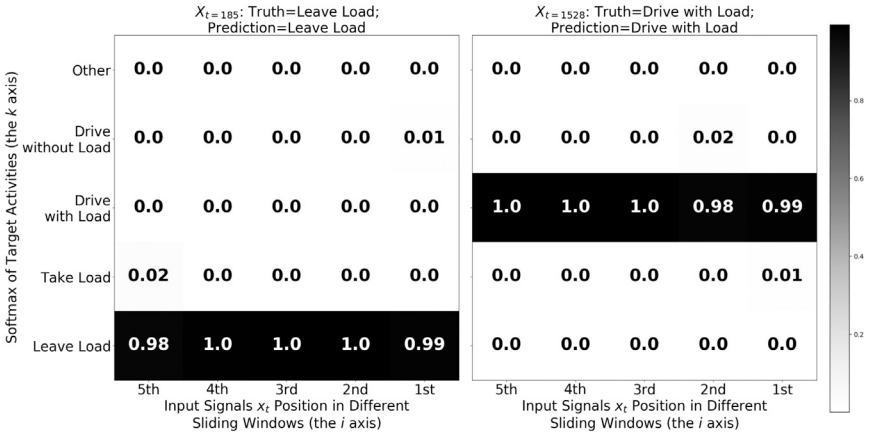

Fig. 4. Two examples of situations with strong context ensemble for *non-overlapping* activities. In the left panel is the correct activity, Leave Load. In the right panel is the correct activity Drive with Load.

Drive without Load to *Leave Load* is detected, the post-processing adjusts the *Leave Load* into a *Take Load*, based on the belief that the *Drive without Load* is correct. As for the erroneous transitions within driving or load-handling, the post-processing modifies the second activity as an extension of the first activity. Suppose that an illegal transition from a *Drive without Load* to a *Drive with Load* is recognized, the *Drive with Load* is then changed into a *Drive without Load*, i.e. the transition is erased.

4.5. Evaluation metrics

The balanced accuracy (BA) (Luque et al., 2019) and the Matthews correlation coefficient (MCC) (Jurman et al., 2012) metrics are used for the evaluation of the results. This is because measuring accuracy on one class versus the others in a multi-class scenario is generally done in an imbalanced situation, and BA and MCC are then better descriptors of the performances (Table 1 shows the frequencies of the different activity categories in the labeled data).

The balanced accuracy is used to measure model performance for each class separately, and it is calculated by

$$\mathrm{BA} = \frac{\mathrm{TPR} + \mathrm{TNR}}{2}, \tag{3}$$

where TPR and TNR are the true-positive rate and the true-negative rate, respectively. In our multi-class setting, labels for a target activity are considered to be positive, and labels for the rest activities are marked as negative. The range of the BA value is between 0 and 1, and a score of 1 for a class means all instances of that class are recognized correctly. The Matthews correlation coefficient is an overall score for all classes. The MCC for a binary classification problem is defined as

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}, \tag{4}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. We use the MCC as generalized by Gorodkin (2004) for the multi-class classification. Since the range of MCC is between −1 and 1, it cannot be compared with BA directly.

5. Experiments and results

In the experiments, the data is prepared with sliding windows, and each input contains 14 CAN signals from a 5 s window. The selected 14 signals include 2 from steering, 2 from heading, 4 from fork adjustment, 3 from fork position and load, and 3 from vehicle velocity. With a sample rate at 10 Hz, the input dimension is $14 \times 5 \times 10$, i.e., a total of 700. Note that the window size m is 5 because the labeling is done at a resolution of each second. Grid search and 10-fold cross-validation are applied in model selection.

The best baseline-RF from 10 experiments is used to generate pseudo-labels from the unlabeled data set. There are 92,000 windows in the unlabeled data set, and from these 7992 windows are selected as *certain* pseudo-labels to incorporate in the training procedure. The total of 10,929 windows, i.e., the labeled (2937 windows) and the *certain* pseudo-labeled data (7992 windows) are used to train different neural networks. Using the experience from our previous work (Chen et al., 2022), parameters of network structure can be approximately estimated at the beginning, e.g., roughly 4 hidden layers; after a model selection, model structures shown in Table 2 are used in experiments. Each experiment was run 10 times, so the average value and standard deviation were available. Last but not least, dropouts (Srivastava et al., 2014), batch normalization (Ioffe and Szegedy, 2015), and early-stopping were used during training to prevent over-fitting, ensuring that all networks were properly trained.

5.1. Overlapping and non-overlapping activities

The test set is 27 min long, divided into 1600 sliding windows. The windows can be separated into two groups: *overlapping* and *non-overlapping* activities. A window with *non-overlapping* activities is located far away from a transition between activities, where it is reasonable to assume that only one activity is taking place within the window. In contrast, a window with *overlapping* activities is located close to an activity transition, where two activities can (and probably do) occur simultaneously.

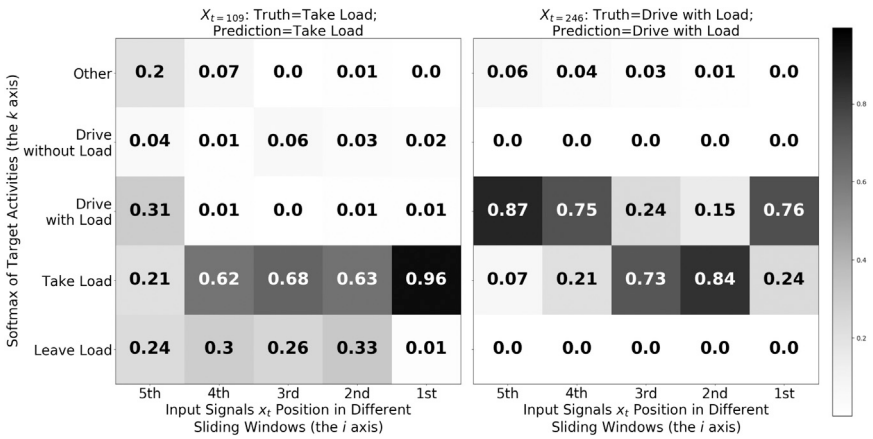

Fig. 5. Two examples of situations with medium context ensemble for *non-overlapping* activities. In the left panel is the correct activity, Take Load. In the right panel is the correct activity Drive with Load.

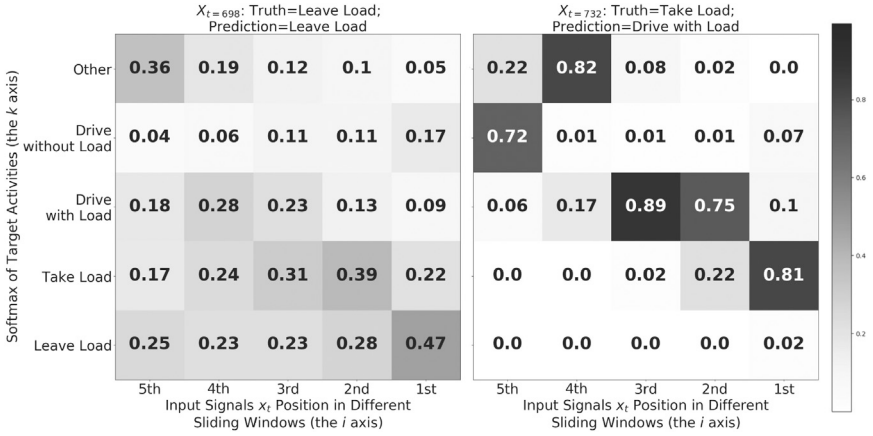

Fig. 6. Two examples of situations with weak context ensemble for *non-overlapping* activities. In the left panel is the correct activity, Leave Load. In the right panel is the correct activity Take Load.

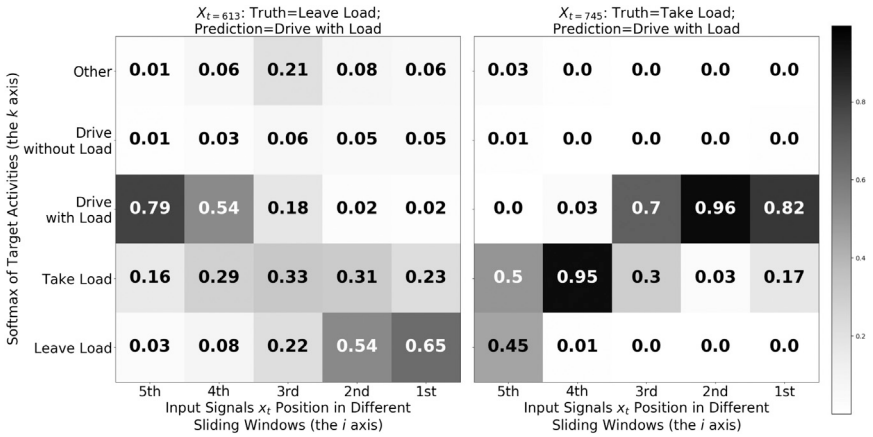

Fig. 7. Two examples of the context ensemble for rapid transitions.

Table 1 shows that the shortest time for an activity in the data is two seconds. We, therefore, consider data that is two seconds or more away from an activity transition (where the expert defined ground-truth label changes) to be *non-overlapping*. There are 1416 windows with *non-overlapping* activity in the 27 min test set. This corresponds to 88.5% of the total number of windows in the test set. Consequently, there are 184 windows with *overlapping* activity, which corresponds to 11.5% of the windows in the test set.

For *non-overlapping* activities, the context ensemble members are often certain about the activity, but sometimes they are quite uncertain. This is illustrated in Figs. 4–6. More than half, 62.64%, of the windows with *non-overlapping* activities have strong context ensembles like those shown in Fig. 4. More than one-third, 34.53%, of the windows with *non-overlapping* activities have medium context ensembles like those shown in Fig. 5. A small part, 2.83%, of the windows with *non-overlapping* activities, have weak context ensembles like those in Fig. 6. The context

Table 3
Classification results on *non-overlapping* and *overlapping* activities.

Activity	Model	Balanced accuracy					MCC*
		Other	Drive without load	Drive with load	Take load	Leave load	All activities
Non-overlapping	Baseline-RF	0.92 ± 0.01	0.89 ± 0.00	0.79 ± 0.01	0.79 ± 0.01	0.69 ± 0.02	0.62 ± 0.01
	MLP	0.92 ± 0.02	0.90 ± 0.01	0.76 ± 0.02	0.78 ± 0.02	0.70 ± 0.03	0.61 ± 0.02
	MLP-CE	0.97 ± 0.01	0.92 ± 0.01	0.79 ± 0.01	0.82 ± 0.02	0.70 ± 0.03	0.66 ± 0.02
	GRU-CE	0.96 ± 0.02	0.92 ± 0.02	0.79 ± 0.02	0.82 ± 0.03	0.71 ± 0.05	0.66 ± 0.03
Overlapping	Baseline-RF	0.74 ± 0.00	0.72 ± 0.02	0.78 ± 0.02	0.57 ± 0.02	0.54 ± 0.02	0.33 ± 0.03
	MLP	0.72 ± 0.06	0.73 ± 0.03	0.71 ± 0.03	0.57 ± 0.03	0.54 ± 0.04	0.29 ± 0.03
	MLP-CE	0.66 ± 0.10	0.84 ± 0.03	0.83 ± 0.01	0.65 ± 0.03	0.51 ± 0.02	0.45 ± 0.03
	GRU-CE	0.65 ± 0.09	0.83 ± 0.03	0.83 ± 0.03	0.65 ± 0.05	0.52 ± 0.03	0.44 ± 0.04

*MCC is the Matthews correlation coefficient in Section 4.

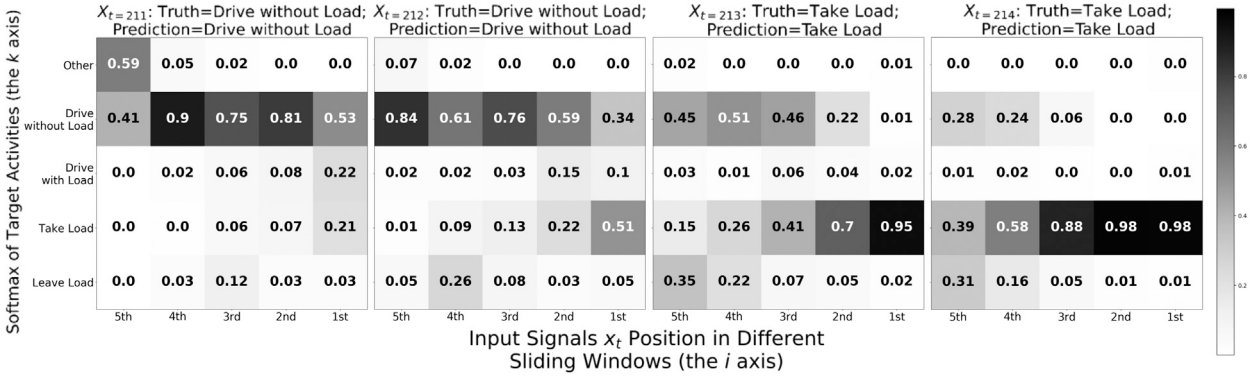

Fig. 8. An example of the context ensemble for long transitions.

ensemble gets the correct answer in most cases with strong and medium context ensembles but often fails in weak cases.

The overall classification scores for *non-overlapping* activities are shown in the upper part of Table 3. The context ensemble improves the recognition results marginally, i.e., very little, compared to the baseline. This is probably to be expected since the *non-overlapping* activity windows are when the differences and similarities between activities should be the clearest. The bottom part of Table 3 shows the overall classification performances on *overlapping* activities. In this case, it is clear that the context ensemble provides a significant improvement compared to the baseline. This is gratifying since it is in the transition periods that we expect to get the most benefit from the context.

For *overlapping* activities, analysis of the context ensemble confidence shows that 19.02% of them are strong, 66.85% of them are medium, and 14.13% of them are weak. Fig. 7 shows two examples of the context ensemble during rapid transitions. The left panel shows a transition from *Drive with Load* to *Leave Load*, and the context ensemble values make sense; when the window looks backward in time, the prediction is *Drive with Load*, and when the window looks forward in time, the prediction is *Leave Load*. The right panel shows a transition from *Take Load* to *Drive with Load*. Similarly, the context ensemble values make sense; when the window looks backward in time, the prediction is *Take Load* (almost confused with *Leave Load*), and when the window looks forward in time, the prediction is *Drive with Load*. An example of a long activity transition is illustrated in Fig. 8, which shows the context ensembles for four consecutive times. This is an *overlapping* activity with a transition from *Drive without Load* to *Take Load*; the transition takes place between the second and the third panel. The context ensemble successfully produces accurate predictions during the whole transition.

5.2. A useful 3D representation

A 3D representation can be extracted from the GRU-CE network's fourth hidden layer with three neurons (see Table 2). This representation is useful for showing how the forklift moves between activities and checking that transitions between activities look sensible. As an illustration, the 58 min data set is fed into the GRU-CE network and plotted with the outputs from the fourth hidden layer in Figs. 9. Generally, all five activities form their own regions; the two load-handling activities are close, and the two driving and the *Other* activities are near to each other.

Fig. 9 also shows the start and end points for the activity *Drive with Load*. Start points are marked with squares and endpoints with triangles. The start and end points are *overlapping* activities, and the start points are often located in between *Take Load* and *Drive with Load*. Similarly, the endpoints tend to be located between *Drive with Load* and *Leave Load*.

5.3. Results after post-processing

The prediction results from the models are finally run through the post-processing based on the logical order of activities, and Table 4 shows the classification results after the post-processing. Consistently, the GRU-CE yields better final results than the other methods, including the MLP with context ensembles. This shows that both the GRU and the context ensemble contribute to the final result.

Getting the proper result after post-processing is non-trivial. Applying the activity logic-based post-processing can lead to several erroneous corrections if it gets offset. Results after post-processing can end up being worse than before, as it seems from comparing Table 3 to Table 4. It is therefore important that the predictions before post-processing are smooth and do not jump between categories, in addition to being as correct as possible. Fig. 10 shows a comparison between ground truth, the baseline-RF recognitions, the GRU-CE recognitions, and the final result after post-processing the GRU-CE recognitions. After post-processing, all eleven working cycles in the test set are recovered. A total of 17.4% of the GRU-CE predictions are modified in the post-processing, which corrects several *Take Load* to *Leave Load* in the first half of the 27 min test set. However, the most interesting case is the "invisible load" at around 1,000 s. Here, the weight is too light to be detected by the forklift weight sensor, so it looks like *Drive without*

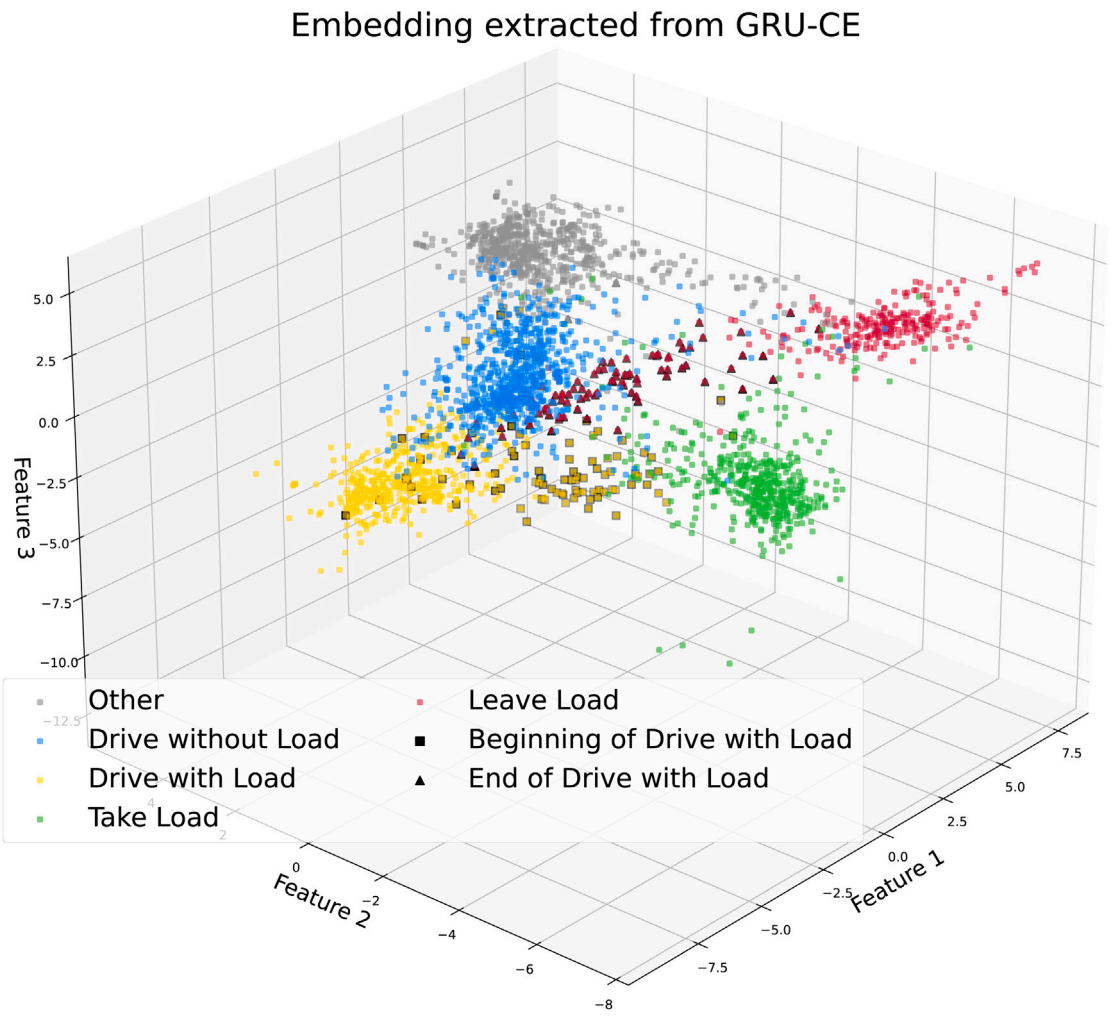

Fig. 9. The 58 min data set plotted in the 3D GRU-CE representation. The squares and triangles mark the start and end points of the activity *Drive with Load*.

Table 4
Classification results after post-processing. "First half" means the first half of the test set with frequent and regular by-the-book operations. "Second half" means the second half with a bit more flexible operations. Boldface indicates the best performance (statistically significant).

Data part	Model	Balanced accuracy					MCC*
		Other	Drive without load	Drive with load	Take load	Leave load	All activities
First half	Baseline-RF	–	0.78 ± 0.01	0.61 ± 0.07	0.84 ± 0.02	0.69 ± 0.02	0.47 ± 0.04
	MLP	–	0.55 ± 0.32	0.55 ± 0.27	0.45 ± 0.32	0.49 ± 0.34	0.00 ± 0.61
	MLP-CE	–	0.72 ± 0.59	0.71 ± 0.52	0.65 ± 0.56	0.66 ± 0.58	0.34 ± 1.09
	GRU-CE	–	**0.94 ± 0.07**	**0.91 ± 0.03**	**0.89 ± 0.02**	**0.90 ± 0.01**	**0.78 ± 0.02**
Second half	Baseline-RF	0.92 ± 0.01	0.20 ± 0.00	0.15 ± 0.01	0.47 ± 0.00	0.47 ± 0.01	−0.37 ± 0.01
	MLP	0.90 ± 0.02	0.46 ± 0.50	0.44 ± 0.48	0.57 ± 0.23	0.60 ± 0.26	0.06 ± 0.77
	MLP-CE	0.94 ± 0.02	0.66 ± 0.78	0.64 ± 0.81	0.66 ± 0.31	0.72 ± 0.40	0.39 ± 1.26
	GRU-CE	0.93 ± 0.03	**0.97 ± 0.02**	**0.97 ± 0.02**	**0.76 ± 0.04**	**0.86 ± 0.14**	**0.89 ± 0.05**

*MCC is the Matthews correlation coefficient in Section 4.5.
– The activity *Other* does not happen in the first half of the test set, therefore, its score is not applicable.

Load. Without considering the longer context and the logical order of activities, it is practically impossible to recognize it correctly as *Drive with Load*.

6. Discussion

Our results show that the stability of model predictions on *overlapping* activities is essential if the order of activities is used in post-processing to get better MAR results. Early and lagging transitions, such as the predictions in Fig. 7, are acceptable for post-processing, as shown in Table 3. However, when the predictions on *overlapping* activities are unstable, i.e., switching back and forth between the two adjacent activities, the post-processing is likely to propagate and amplify these errors. Fig. 11 shows an example of post-processing unstable predictions during a fuzzy transition, and the rigid rule-based post-processing forces every transition to match. This is problematic when handling unstable predictions; for example, it turns the second *Drive with Load* into *Drive without Load*, based on the fact that the first *Leave Load* activity has been revised. Furthermore, this erroneous adjustment propagates to all predictions coming after it, creating more errors. Thus, the outcome after post-processing is a measure of the stability of the model — it is effective only with models that provide stable predictions

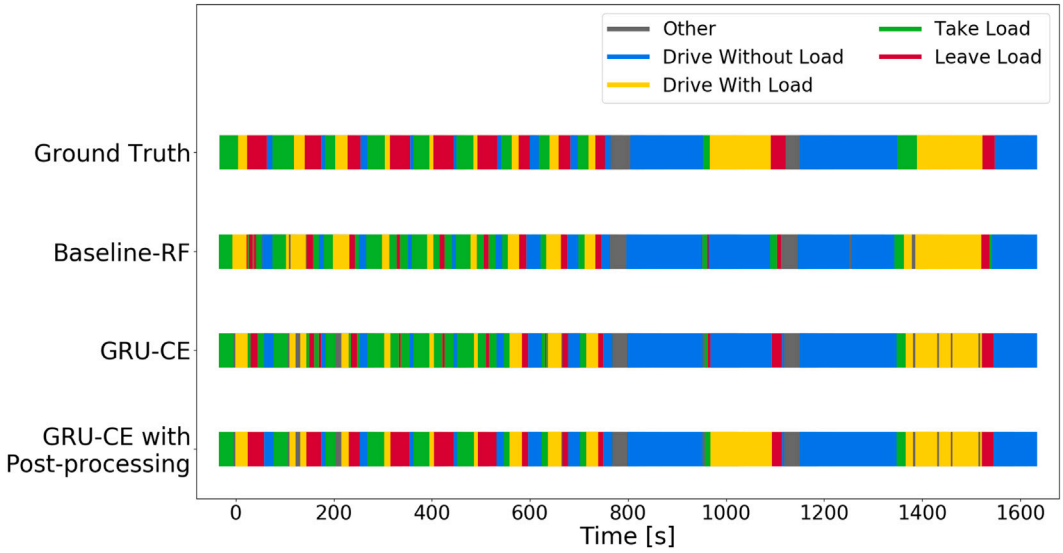

Fig. 10. Ground truth and predictions from different experiments on the test set. The post-processing forces the activities to follow the color logic: blue → green → yellow → red → blue, and so on.

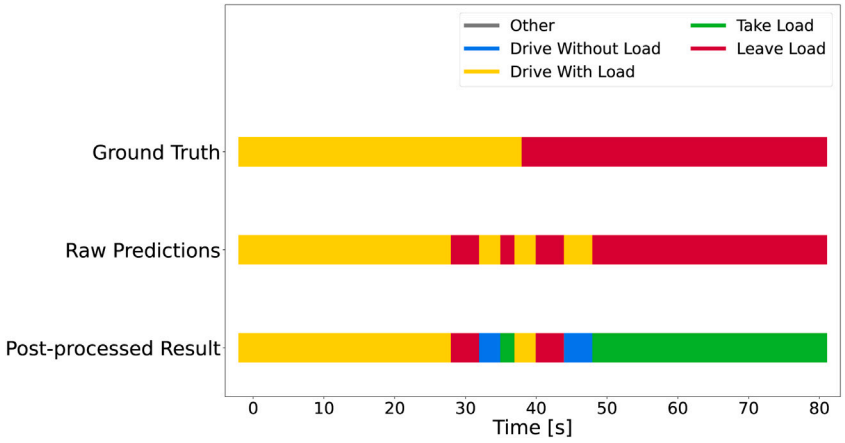

Fig. 11. An example of how instability during the transition has a negative influence on the recognition result after post-processing. The post-processing adjusts the activities to follow the logic sequence of the activities, here illustrated with color: blue → green → yellow → red → blue, and so on.

on *overlapping* activities; in our case, only the GRU context ensemble method is stable.

To further understand the stability of the proposed context ensemble, two more series of experiments are conducted; the corresponding results are shown in Tables B.5 and B.6 in Appendix B. The window size m is set to 5 in both series of experiments. Experiments of the basic concept of the ensemble are denoted as RF-E, MLP-E, and GRU-E (where E is short for "ensemble"). In the first series of experiments, models are trained with random subsets of the training data to predict the last activity in a sliding window, i.e., the y_t in W_t. The committee consists of predictions from all these models, and a majority vote is used to decide the final prediction. The results of this series of experiments are identical to the baseline experiments, where RFs are trained with only 58 min of labeled data. Thus, this traditional form of ensemble methods does not provide any improvement.

The second series of experiments are similar to the proposed context ensemble. The notation is RF-C, MLP-C, and GRU-C where C stands for context. For an experiment setting where the window size is m, m models are trained with the same input, i.e., W_t, but to handle different context in the window: one model predicting y_t, another model predicting y_{t-1}, and so on until the last model predicting y_{t-m+1}. Thus, instead of using one model, we train different models for different time steps. As the window is moving forward by 1 timestep, for each input x_t, a committee similar to the one in Fig. 2 is formed, and a majority vote is applied. In this case, predictions from different contexts are derived from different individual models (instead of being derived from one single model in the proposed context ensemble). Results show that models in this series of experiments show similar classification performance on *overlapping* activities as the context ensemble method. However, it has poorer stability and fails to deliver decent MAR results after post-processing. Furthermore, even if they have similar final results, the proposed context ensemble holds the advantage of training one single model. From a multi-task training perspective, this should be beneficial.

7. Conclusion

Successful productivity monitoring relies on accurate activity recognition results. To develop practical solutions for the forklift MAR problem, one needs to overcome the challenges of fuzzy activity transition caused by flexible operations. In this paper, we proposed a GRU-based method together with context ensembles to recognize five basic forklift

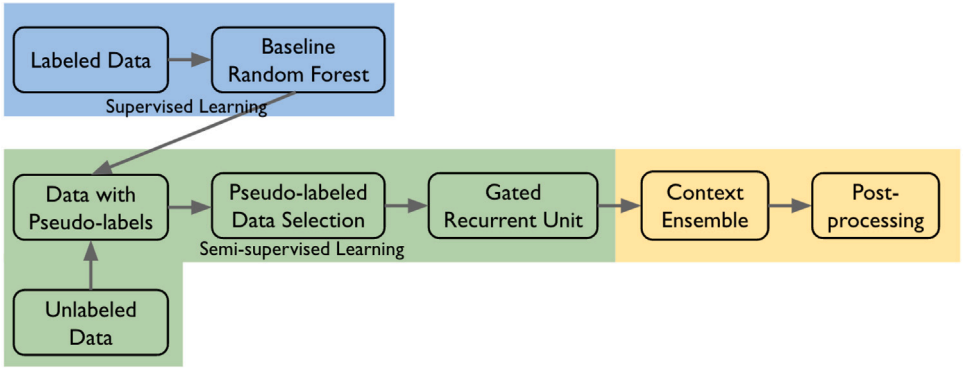

Fig. A.12. Flow Chart of the Methodology.

activities. The results show that in more than 95% of the cases, medium or strong confident context ensembles can be formed to provide robust activity predictions. Using context ensemble with neural network-based algorithms can obtain significant improvement on most of the *overlapping driving* and *load-handling* activities and have a decent performance on all *non-overlapping* activities. Moreover, the proposed GRU-CE significantly improves classification performance during activity transitions and prepares stable predictions for proper post-processing. The final results show that the GRU-CE outperforms other approaches in both by-the-book and more flexible usage, achieving a balanced accuracy of 97% for driving activities and 90% for load-handling activities. Last but not least, transitions between forklift activities can be observed and intuitively understood in the embedding extracted from the GRU network, which verifies that the model captures the knowledge of activity and activity transition.

An important future work is to improve the methods to capture long-term context so that the post-processing step can be removed. One reason is that the prerequisites of the current post-processing are very rigid. More flexible forklift usage, such as stacking pallets, remains challenging to recognize. Other than that, it would also be interesting to see whether the idea of GRUs and context ensembles applied to CAN-data can be effective in other MAR applications, for example, construction equipment.

CRediT authorship contribution statement

Kunru Chen: Conceptualization, Methodology, Validation, Data curation, Formal analysis, Software, Visualization, Writing – original draft, Writing – review & editing. **Thorsteinn Rögnvaldsson:** Conceptualization, Methodology, Validation, Funding acquisition, Project administration, Supervision, Writing – review & editing. **Sławomir Nowaczyk:** Methodology, Validation, Writing – original draft, Funding acquisition, Supervision, Writing – review & editing. **Sepideh Pashami:** Methodology, Supervision, Writing – review & editing. **Jonas Klang:** Project administration, Supervision. **Gustav Sternelöv:** Resource gathering.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jonas Klang and Gustav Sternelov reports financial support was provided by Toyota Material Handling Europe.

Data availability

The data that has been used is confidential.

Acknowledgments

We would like to express our gratitude to Toyota Material Handling Manufacturing Sweden AB and Stiftelsen för Kunskaps-och Kompetensutveckling (SE) for support for this research. All authors have read and agreed to the published version of the manuscript.

List of Abbreviations.

BA	Balanced accuracy
CAN	Controller Area Network
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-term Memory
MAR	Machine Activity Recognition
MCC	Matthews Correlation Coefficient
RF	Random Forest
RNN	Recurrent Neural Network

Appendix A. Other details of the proposed method

In this section, we present more details about the proposed method, which consists of the following six steps (also shown in Fig. A.12):

1. **Baseline-RF:** the 58 min of labeled data is used to train a random forest model, denoted as baseline-RF;
2. **Pseudo-labeled data generation:** the baseline-RF is applied on the 25 h of unlabeled data, obtaining 25 h of pseudo-labeled data;
3. **Pseudo-labeled data selection:** as all labels in w_t are needed to train the GRU network in the proposed context ensemble method, predictions for every timestep in a window, i.e., from $\mathbf{x}_{t-m+1}$ to $\mathbf{x}_t$ in w_t, are collected. For windows where all predictions belong to the same activity (the window is most likely a *non-overlapping activity*), it is selected to be used in the next step. For windows where predictions consist of two activities, the first and the last pseudo-labels are checked, and if these two labels follow the correct transition (defined in Section 4.4), the window is selected for the next step. As for the windows where three or more activities are found in their pseudo-labels, they are not used further as their pseudo-labels are quite noisy;
4. **Training the GRU network:** the network is trained with a mix of 58 min labeled data and the selected pseudo-labeled data in the previous step;
5. **Context ensemble:** all details about the context ensemble are presented in Section 4.3;
6. **Post-processing:** all details about the rule-based post-processing are presented in Section 4.4.

Table B.5
Classification results on *non-overlapping* and *overlapping* activities.

Activity	Model	Balanced accuracy					MCC*
		Other	Drive without load	Drive with load	Take load	Leave load	All activities
Non-overlapping	Baseline-RF	0.92 ± 0.01	0.89 ± 0.00	0.79 ± 0.01	0.79 ± 0.01	0.69 ± 0.02	0.62 ± 0.01
	MLP	0.92 ± 0.02	0.90 ± 0.01	0.76 ± 0.02	0.78 ± 0.02	0.70 ± 0.03	0.61 ± 0.02
	RF-E	0.93 ± 0.01	0.88 ± 0.01	0.78 ± 0.01	0.76 ± 0.02	0.55 ± 0.04	0.55 ± 0.02
	MLP-E	0.94 ± 0.01	0.90 ± 0.01	0.76 ± 0.01	0.75 ± 0.01	0.70 ± 0.02	0.60 ± 0.01
	GRU-E	0.94 ± 0.01	0.90 ± 0.00	0.76 ± 0.01	0.76 ± 0.01	0.69 ± 0.01	0.60 ± 0.01
	RF-C	0.96 ± 0.00	0.91 ± 0.01	0.80 ± 0.01	0.82 ± 0.01	0.62 ± 0.01	0.63 ± 0.01
	MLP-C	0.97 ± 0.01	0.92 ± 0.00	0.79 ± 0.01	0.82 ± 0.01	0.71 ± 0.01	0.67 ± 0.00
	GRU-C	0.96 ± 0.01	0.92 ± 0.00	0.78 ± 0.01	0.81 ± 0.03	0.71 ± 0.02	0.65 ± 0.02
	MLP-CE	0.97 ± 0.01	0.92 ± 0.01	0.79 ± 0.01	0.82 ± 0.02	0.70 ± 0.03	0.66 ± 0.02
	GRU-CE	0.96 ± 0.02	0.92 ± 0.02	0.79 ± 0.02	0.82 ± 0.03	0.71 ± 0.05	0.66 ± 0.03
Overlapping	Baseline-RF	0.74 ± 0.00	0.72 ± 0.02	0.78 ± 0.02	0.57 ± 0.02	0.54 ± 0.02	0.33 ± 0.03
	MLP	0.72 ± 0.06	0.73 ± 0.03	0.71 ± 0.03	0.57 ± 0.03	0.54 ± 0.04	0.29 ± 0.03
	RF-E	0.55 ± 0.05	0.73 ± 0.01	0.82 ± 0.02	0.53 ± 0.03	0.49 ± 0.00	0.30 ± 0.02
	MLP-E	0.63 ± 0.06	0.75 ± 0.06	0.82 ± 0.03	0.53 ± 0.03	0.49 ± 0.01	0.32 ± 0.05
	GRU-E	0.60 ± 0.01	0.74 ± 0.03	0.83 ± 0.03	0.52 ± 0.03	0.50 ± 0.01	0.31 ± 0.03
	RF-C	0.73 ± 0.05	0.83 ± 0.02	0.82 ± 0.01	0.60 ± 0.03	0.50 ± 0.01	0.43 ± 0.02
	MLP-C	0.66 ± 0.09	0.84 ± 0.01	0.83 ± 0.02	0.64 ± 0.02	0.53 ± 0.01	0.46 ± 0.02
	GRU-C	0.65 ± 0.06	0.84 ± 0.02	0.82 ± 0.02	0.63 ± 0.03	0.53 ± 0.04	0.44 ± 0.04
	MLP-CE	0.66 ± 0.10	0.84 ± 0.03	0.83 ± 0.01	0.65 ± 0.03	0.51 ± 0.02	0.45 ± 0.03
	GRU-CE	0.65 ± 0.09	0.83 ± 0.03	0.83 ± 0.03	0.65 ± 0.05	0.52 ± 0.03	0.44 ± 0.04

*MCC is the Matthews correlation coefficient in Section 4.

Table B.6
Classification results after post-processing. "First half" means the first half of the test set with frequent and regular by-the-book operations. "Second half" means the second half with a bit more flexible operations. Boldface indicates the best performance (statistically significant).

Data part	Model	Balanced accuracy					MCC*
		Other	Drive without load	Drive with load	Take load	Leave load	All activities
First half	Baseline-RF	–	0.78 ± 0.01	0.61 ± 0.07	0.84 ± 0.02	0.69 ± 0.02	0.47 ± 0.04
	MLP	–	0.55 ± 0.32	0.55 ± 0.27	0.45 ± 0.32	0.49 ± 0.34	0.00 ± 0.61
	RF-E	–	0.81 ± 0.25	0.83 ± 0.20	0.80 ± 0.12	0.83 ± 0.12	0.62 ± 0.30
	MLP-E	–	0.45 ± 0.09	0.51 ± 0.06	0.38 ± 0.05	0.40 ± 0.06	−0.15 ± 0.10
	GRU-E	–	0.72 ± 0.36	0.70 ± 0.33	0.61 ± 0.34	0.66 ± 0.37	0.31 ± 0.67
	RF-C	–	0.73 ± 0.56	0.74 ± 0.48	0.68 ± 0.48	0.69 ± 0.48	0.40 ± 0.96
	MLP-C	–	0.48 ± 0.48	0.49 ± 0.43	0.41 ± 0.46	0.43 ± 0.47	−0.10 ± 0.89
	GRU-C	–	0.72 ± 0.59	0.69 ± 0.50	0.65 ± 0.56	0.67 ± 0.57	0.34 ± 1.07
	MLP-CE	–	0.72 ± 0.59	0.71 ± 0.52	0.65 ± 0.56	0.66 ± 0.58	0.34 ± 1.09
	GRU-CE	–	**0.94 ± 0.07**	**0.91 ± 0.03**	**0.89 ± 0.02**	**0.90 ± 0.01**	**0.78 ± 0.02**
Second half	Baseline-RF	0.92 ± 0.01	0.20 ± 0.00	0.15 ± 0.01	0.47 ± 0.00	0.47 ± 0.01	−0.37 ± 0.01
	MLP	0.90 ± 0.02	0.46 ± 0.50	0.44 ± 0.48	0.57 ± 0.23	0.60 ± 0.26	0.06 ± 0.77
	RF-E	0.89 ± 0.01	0.73 ± 0.56	0.71 ± 0.60	0.69 ± 0.23	0.70 ± 0.27	0.49 ± 0.91
	MLP-E	0.91 ± 0.01	0.43 ± 0.40	0.40 ± 0.36	0.56 ± 0.19	0.63 ± 0.26	0.02 ± 0.61
	GRU-E	0.91 ± 0.01	0.74 ± 0.42	0.70 ± 0.35	0.67 ± 0.18	0.73 ± 0.27	0.52 ± 0.65
	RF-C	0.93 ± 0.01	0.66 ± 0.65	0.65 ± 0.66	0.69 ± 0.29	0.67 ± 0.34	0.39 ± 1.05
	MLP-C	0.95 ± 0.02	0.35 ± 0.45	0.31 ± 0.45	0.57 ± 0.18	0.64 ± 0.19	−0.09 ± 0.72
	GRU-C	0.94 ± 0.01	0.51 ± 0.55	0.46 ± 0.52	0.59 ± 0.21	0.66 ± 0.29	0.16 ± 0.90
	MLP-CE	0.94 ± 0.02	0.66 ± 0.78	0.64 ± 0.81	0.66 ± 0.31	0.72 ± 0.40	0.39 ± 1.26
	GRU-CE	0.93 ± 0.03	**0.97 ± 0.02**	**0.97 ± 0.02**	**0.76 ± 0.04**	**0.86 ± 0.14**	**0.89 ± 0.05**

*MCC is the Matthews correlation coefficient in Section 4.5.
– The activity *Other* does not happen in the first half of the test set, therefore, its score is not applicable.

Appendix B. More comparison on ensemble methods

In this section, we present the results from two more experiments mentioned in Section 6: Table B.5 is an extension of Table 3, and Table B.6 is an extension of Table 4. Experiment details of baseline-RF, MLP, MLP-CE and GRU-CE are presented in Section 5, and details of the model-E and model-C experiments are explained in Section 6.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2023.106992.

References

Ahn, C.R., Lee, S., Peña-Mora, F., 2015. Application of low-cost accelerometers for measuring the operational efficiency of a construction equipment fleet. J. Comput. Civ. Eng. 29, 04014042.

Akhavian, R., Behzadan, A.H., 2015. Construction equipment activity recognition for simulation input modeling using mobile sensors and machine learning classifiers. Adv. Eng. Inform. 29, 867–877.

Akinosho, T.D., Oyedele, L.O., Bilal, M., Ajayi, A.O., Delgado, M.D., Akinade, O.O., Ahmed, A.A., 2020. Deep learning in the construction industry: A review of present status and future innovations. J. Build. Eng. 32, 101827.

Ali Hamad, R., Salguero Hidalgo, A., Bouguelia, M.R., Espinilla Estevez, M., Medina Quero, J., 2020. Efficient activity recognition in smart homes using delayed fuzzy temporal windows on binary sensors. IEEE J. Biomed. Health Inform. 24, 387–395.

Chen, K., Rögnvaldsson, T., Nowaczyk, S., Pashami, S., Johansson, E., Sternelöv, G., 2022. Semi-supervised learning for forklift activity recognition from Controller Area Network (CAN) signals. Sensors 22 (11), http://dx.doi.org/10.3390/s22114170.

Chen, C., Zhu, Z., Hammad, A., 2020. Automated excavators activity recognition and productivity analysis from construction site surveillance videos. Autom. Constr. 110, 103045. http://dx.doi.org/10.1016/j.autcon.2019.103045.

Cheng, C.F., Rashidi, A., Davenport, M.A., Anderson, D.V., 2017. Activity analysis of construction equipment using audio signals and support vector machines. Autom. Constr. 81, 240–253.

Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. http://dx.doi.org/10.48550/ARXIV.1409.1259, arXiv.

Elman, J.L., 1990. Finding structure in time. Cogn. Sci. 14 (2), 179–211. http://dx.doi.org/10.1016/0364-0213(90)90002-E.

Fischer, A., Bedrikow, A.B., Kessler, S., Fottner, J., 2021a. Equipment data-based activity recognition of construction machinery. In: 2021 IEEE International Conference on Engineering, Technology and Innovation. ICE/ITMC, pp. 1–6. http://dx.doi.org/10.1109/ICE/ITMC52061.2021.9570272.

Fischer, A., Liang, M., Orschlet, V., Bi, H., Kessler, S., Fottner, J., 2021b. Detecting equipment activities by using machine learning algorithms. IFAC-PapersOnLine 54 (1), 799–804. http://dx.doi.org/10.1016/j.ifacol.2021.08.094, URL: https://www.sciencedirect.com/science/article/pii/S2405896321008405, 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021.

Golparvar-Fard, M., Heydarian, A., Niebles, J.C., 2013. Vision-based action recognition of earthmoving equipment using spatio-temporal features and support vector machine classifiers. Adv. Eng. Inform. 27, 652–663.

Gong, J., Caldas, C.H., 2009. An intelligent video computing method for automated productivity analysis of cyclic construction operations. In: International Workshop on Computing in Civil Engineering 2009. pp. 64–73.

Gorodkin, J., 2004. Comparing two K-category assignments by a K-category correlation coefficient. Comput. Biol. Chem. 28 (5), 367–374. http://dx.doi.org/10.1016/j.compbiolchem.2004.09.006.

Harichandran, A., Raphael, B., Mukherjee, A., 2021. A hierarchical machine learning framework for the identification of automated construction operations. J. Inf. Technol. Constr. 26, 591–623.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. http://dx.doi.org/10.48550/ARXIV.1502.03167, arXiv, URL: https://arxiv.org/abs/1502.03167.

Jakobsson, E., Frisk, E., Krysander, M., Pettersson, R., 2020. Automated usage characterization of mining vehicles for life time prediction. IFAC-PapersOnLine 53 (2), 11950–11955.

Jung, S., Jeoung, J., Kang, H., Hong, T., 2021. 3D convolutional neural network-based one-stage model for real-time action detection in video of construction equipment. Comput.-Aided Civ. Infrastruct. Eng. 1–17.

Jurman, G., Riccadonna, S., Furlanello, C., 2012. A comparison of MCC and CEN error measures in multi-class prediction. PLoS One 7, e41882.

Kim, H., Ahn, C.R., Engelhaupt, D., Lee, S., 2018. Application of dynamic time warping to the recognition of mixed equipment activities in cycle time measurement. Autom. Constr. 87, 225–234.

Kim, J., Chi, S., 2019. Action recognition of earthmoving excavators based on sequential pattern analysis of visual features and operation cycles. Autom. Constr. 104, 255–264.

Kim, J., Chi, S., 2020. Multi-camera vision-based productivity monitoring of earthmoving operations. Autom. Constr. 112, 103121.

Kolkar, R., Singh Tomar, R.P., Vasantha, G., 2022. IoT-based human activity recognition models based on CNN, LSTM and GRU. In: 2022 IEEE Silchar Subsection Conference. SILCON, pp. 1–7. http://dx.doi.org/10.1109/SILCON55242.2022.10028803.

Lee, D.h., 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks.

Luque, A., Carrasco, A., Martín, A., de las Heras, A., 2019. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. Pattern Recognit. 91, 216–231.

Mahami, H., Nasirzadeh, F., Hosseininaveh Ahmadabadian, A., Esmaeili, F., Nahavandi, S., 2019. Imaging network design to improve the automated construction progress monitoring process. Constr. Innov. 19 (3), 386–404. http://dx.doi.org/10.1108/CI-07-2018-0059.

Mohsen, S., 2023. Recognition of human activity using GRU deep learning algorithm. Multimedia Tools Appl. http://dx.doi.org/10.1007/s11042-023-15571-y.

Rashid, K.M., Louis, J., 2020a. Activity identification in modular construction using audio signals and machine learning. Autom. Constr. 119, 103361. http://dx.doi.org/10.1016/j.autcon.2020.103361.

Rashid, K.M., Louis, J., 2020b. Automated activity identification for construction equipment using motion data from articulated members. Front. Built Environ. 5, 144.

Sherafat, B., Ahn, C.R., Akhavian, R., Behzadan, A.H., Golparvar-Fard, M., Kim, H., Lee, Y.C., Rashidi, A., Azar, E.R., 2020. Automated methods for activity recognition of construction workers and equipment: State-of-the-art review. J. Constr. Eng. Manag. 146, 03120002.

Shi, Y., Xia, Y., Zhang, Y., Yao, Z., 2020. Intelligent identification for working-cycle stages of excavator based on main pump pressure. Autom. Constr. 109, 102991.

Shiri, F.M., Perumal, T., Mustapha, N., Mohamed, R., Ahmadon, M.A.B., Yamaguchi, S., 2023. A survey on multi-resident activity recognition in smart environments. arXiv:2304.12304.

Slaton, T., Hernandez, C., Akhavian, R., 2020. Construction activity recognition with convolutional recurrent networks. Autom. Constr. 113, 103138.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.

Sun, X., Xu, H., Dong, Z., Shi, L., Liu, Q., Li, J., Li, T., Fan, S., Wang, Y., 2022. CapsGaNet: Deep neural network based on capsule and GRU for human activity recognition. IEEE Syst. J. 16 (4), 5845–5855. http://dx.doi.org/10.1109/JSYST.2022.3153503.

Wu, Y., Wang, M., Liu, X., Wang, Z., Ma, T., Lu, Z., Liu, D., Xie, Y., Li, X., Wang, X., 2021. Monitoring the work cycles of earthmoving excavators in earthmoving projects using UAV remote sensing. Remote Sens. 13 (19), http://dx.doi.org/10.3390/rs13193853.