IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 14, 2020, accepted December 23, 2020, date of publication January 14, 2021, date of current version January 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051763

Clustered Multi-Task Sequence-to-Sequence Learning for Autonomous Vehicle Repositioning

SANGMIN LEE[1], DAE-EUN LIM[2], YOUNKOOK KANG[3], AND HAE JOONG KIM[3]

[1]School of Information Convergence, Kwangwoon University, Seoul 01897, South Korea
[2]Department of Industrial Engineering, Kangwon National University, Chuncheon 24341, South Korea
[3]Material Handling Automation Group, Samsung Electronics, Hwaseong 445-330, South Korea

Corresponding author: Hae Joong Kim (haejoong.kim@gmail.com)

The present research has been conducted by the Research Grant of Kwangwoon University in 2020. This research has been supported by Samsung Electronics, Co., Ltd. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2020R1I1A3A04037238). This study has been worked with the support of a research grant of Kangwon National University in 2020.

ABSTRACT Clustered multi-task learning, which aims to leverage the generalization performance over clustered tasks, has shown an outstanding performance in various machine learning applications. In this paper, a clustered multi-task sequence-to-sequence learning (CMSL) for autonomous vehicle systems (AVSs) in large-scale semiconductor fabrications (fab) is proposed, where AVSs are widely used for wafer transfers. Recently, as fabs become larger, the repositioning of idle vehicles to where they may be requested has become a significant challenge because inefficient vehicle balancing leads to transfer delays, resulting in production machine idleness. However, existing vehicle repositioning systems are mainly controlled by human operators, and it is difficult for such systems to guarantee efficiency. Further, we should handle the small data problem, which is insufficient for machine learning because of the irregular time-varying manufacturing environments. The main purpose of this study is to examine CMSL-based predictive control of idle vehicle repositioning to maximize machine utilization. We conducted an experimental evaluation to compare the prediction accuracy of CMSL with existing methods. Further, a case study in a real largescale semiconductor plant, demonstrated that the proposed predictive approach outperforms the existing approaches in terms of transfer efficiency and machine utilization.

INDEX TERMS Clustered multi-task learning, sequence-to-sequence, vehicle repositioning, idle vehicle balancing, automated material handling systems, overhead hoist transports.

I. INTRODUCTION

Over the past 30 years, automated material handling systems (AMHSs) have been widely applied in semiconductor plants to guarantee efficient, safe, and fast transportation of wafers between facilities. AMHSs prevent particle contamination and increase the tool utilization, resulting in high productivity by reducing the cycle time. Autonomous vehicle systems (AVSs), the latest version of AMHSs, aim to transfer materials using overhead hoist transports (OHTs) that travel along the railways installed on the ceiling of a fab.

Optimizing the AVS operation is a significant challenge in a large fab where several hundreds of vehicles travel along the railways with dozens of areas. Idle vehicle repositioning is one of the significant control decisions in AVSs because idle vehicle repositioning affects the unload transfer time (UTT) for idle vehicles moving to the pickup point. Although the average loading in an area has been optimized at the fab layout design stage, the transport requests for each area can change over time owing to changes in the product mix, and the fluctuations of the machine capability [1]. The lack of idle vehicles compared with the transfer demands in an area causes severe unload transfer delays, which results in machine idleness. Fig. 1 provides two examples of areas where, although the total number of idle vehicles is sufficient, unload transfer delays occur owing to insufficient idle vehicles. In areas k and m, the skewness value of the unload transfer time distribution increases from week W_{t-2} to W_t although average times are almost constant.

In this paper, we propose a predictive approach to ensuring a robust vehicle repositioning system. The transfer demand, the number of transfer requests for an area, is predicted one-time unit (15 min) in advance. We require a highly accurate predictor because inaccurate prediction results in various side effects, which include a high rate of unload transfer delays, AMHS incapability, and local congestion, resulting in

The associate editor coordinating the review of this manuscript and approving it for publication was Zhe Xiao.

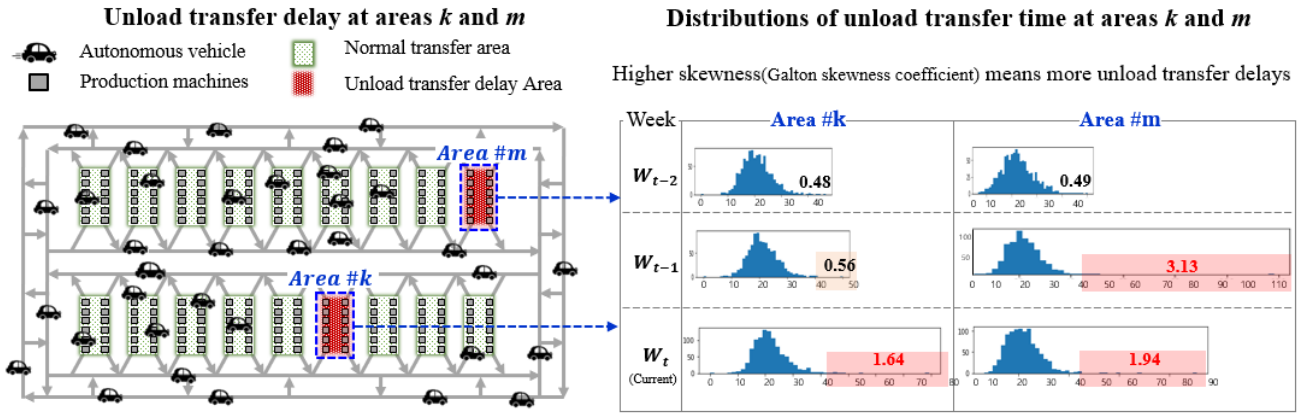


FIGURE 1. Example of unload transfer delay in areas *m* and *k*. These delays, which are caused by changes the transport requests for each area, bring about the inefficiency of the production machines, resulting in production losses.

machine idleness [2], [3]. Regarding the prediction accuracy, we should consider the overfit and underfit problems of a prediction model for each area. To resolve this, clustered sequence-to-sequence multi-task learning is proposed for idle vehicle repositioning in AVSs. Clustered multi-task learning techniques [4] leverage the prediction accuracy by sharing useful information between related tasks, whereas sequence-to-sequence (Seq2Seq) techniques identify the variability of the transfer request demands by identifying the demand sequence information. Further, to illustrate the effectiveness of the proposed approach, we describe a thorough explanation of the overall procedure, which details the data preprocessing and learning framework. For practical solutions, a real-time version of the vehicle repositioning system was implemented and applied to several scalable semiconductor plants.

The main contributions of this study can be summarized as follows. We propose a clustered sequence-to-sequence multi-task learning for autonomous vehicle systems. Clustered multi-task learning aims to leverage learning efficiency and prediction accuracy by sharing useful information between related tasks. Further, sequence-to-sequence techniques identify the variability of the transfer request demands by identifying the autocorrelations of the demand requests. Comparative studies show that in terms of prediction accuracy, the proposed method outperforms the representative methods.

The remainder of this paper is organized as follows. Sections II and III review the relevant literature and preliminaries, respectively. Section IV describes the proposed method, and Section V details the field application. Finally, Section VI provides some concluding remarks.

II. LITERATURE REVIEW

The idle vehicle repositioning problem begins by determining the dwell positions for idle vehicles. Egbelu [5] introduced the idle vehicle positioning problem to minimize the maximum empty travel time from the home position to the load pick-up point in a loop layout. Kim [6] proposed a polynomial-time algorithm for idle vehicles in a loop layout with the objective of minimizing the average response time. A dynamic programming algorithm for positioning idle AGVs in a loop layout was proposed by Gademann and van de Velde [7]. Lee and Ventura [8] also proposed a polynomial-time algorithm based on dynamic programming for the idle vehicle positioning problem in both multiple-vehicle uni- and bi-directional loop layouts to minimize the mean response time. A dynamic programming algorithm for determining an optimal set of dwell points has been extended to a tandem-loop multiple-vehicle AGV system by Lee [9]. However, previous studies have only considered small-sized problems involving a few vehicles and a single or a few loops. Moreover, because they are not based on a data-driven approach, they are vulnerable to environmental uncertainties.

The vehicle fleet sizing problem determines the required number of OHT vehicles in an OHT loop in a segregated AMHS that utilizes inter- and intra-area transport systems separately. Huang *et al.* [10] attempted to find the optimal vehicle allocation for the inter-bay and intra-bay systems using a discrete event simulation model. They adapted the simulation optimization, called convergent optimization, through the most-promising-area stochastic search algorithm, to reduce the number of necessary simulation runs to find an optimal solution. The same problem can be obtained by other approaches, such as that developed by Chang *et al.* [11], who used the simulation sequential metamodeling method based on functional relations between the input and output of a simulation model, including the vehicle amount and the corresponding transport time. The metamodels are built to find the minimum necessary fleet size while fulfilling a production-based process to process time constraints. Other approaches focus on how to reduce the necessary amount of simulation runs to find the optimal fleet size using evolutionary algorithms, as by Lin and Huang [12]. Subsequently, Lin and Huang [13] conducted a similar approach using an optimal computing budget allocation together with particle swarm optimization to find the best combination of

dispatching rules and the number of vehicles to operate a photo area. However, vehicle fleet sizing methods have only focused on determining the number of vehicles for each area, and have not considered the dynamic repositioning of an idle vehicle. Because OHT vehicles travel among loops in a unified transport, a dynamic vehicle allocation problem arises when the requirement of each loop changes over time.

To prevent congestion or idle time in each intrabay, Lin *et al.* [14] proposed a zone control strategy that keeps the number of vehicles in each intrabay between the upper and lower limits. Their results indicate that their strategy significantly affects the performance of AMHS. Several studies investigated whether the idle vehicle circulation policy significantly affects the performance of the AMHS, and recommended a circulation path across the bays to balance the number of idle vehicles among the bays [15], [16]. Lin *et al.* [17] introduced a Markov decision process model to solve the vehicle assignment problem in a dynamic manner. To increase the calculation efficiency, they described a dynamic vehicle allocation control to assign the optimal number of vehicles to each area. Although this approach is quite promising, the example scenario only implies two areas.

Kiba *et al.* [18] investigated a minimum service policy that assigns a minimum number of idle vehicles to bays in order to quickly react to transport requests. This policy defines low water marks (LWMs), which serve as a lower control limit to define the necessary minimum number of vehicles available per area. The study shows that the minimum service policy is more effective than a classical strategy. Additional high-water marks (HWM), mentioned by Jimenez *et al.* [19], serve to prevent the risk of traffic jams. If the HWM value is exceeded, the area pushes out empty vehicles to other areas. Two strategies, area dedication and zone balancing, are compared. The water calculation and area dynamics were first directly addressed by Chaabane *et al.* [20]. Again, a minimum service policy was used that included a degree of freedom to reduce the number of vehicles available for empty vehicle dispatching. LWMs are calculated based on the same proportion as the area transport to total transport, and reduced by the degree of freedom. This approach was extended to include system dynamics (transport leaving versus transport staying in the same area). However, nothing was written about the method of vehicle exchange when the limits were reached. Moreover, using a minimum service policy cannot guarantee an outperformance when the minimum number of vehicles is not allocated owing to a failure to predict the transport requests of each area.

Schmaler *et al.* [21] introduced the possibility of available forecast data into empty vehicle dispatching strategies for the unified AMHS of a semiconductor manufacturing facility. This study shows that the scenario of using only information regarding upcoming tool events, and thus only half of all possible future event information, performs better than a watermark-based method. Ahn and Park [22] proposed a rebalancing strategy for idle OHTs to reduce retrieval time and congestion. They discretized the fab into a number of zones and presented rebalancing strategies for each zone by using multi-agent reinforcement learning. However, these approaches cannot be applied to real-world large-scale plants because they overlook the actual specifications such as the movement of thousands of vehicles and severe changes in the traffic pattern. Moreover, owing to the difficulty of reinforcement learning convergence and application, human intervention is still essential to building well-trained models.

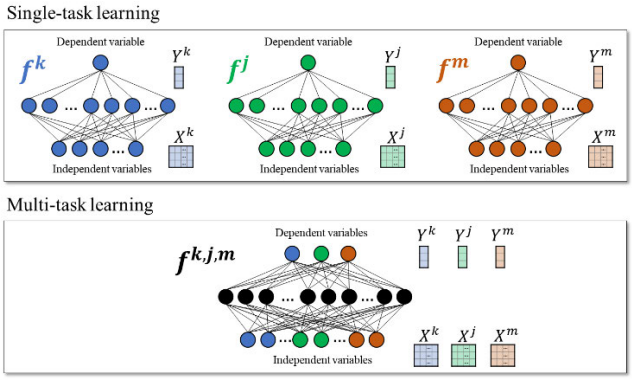


FIGURE 2. Conceptual framework of single- and multi-task learning with neural networks where all the tasks or a subset of tasks are related.

III. PRELIMINARIES

A. MULTI-TASK LEARNING

Multi-task learning is an emerging machine learning approach for learning-related tasks. Fig. 2 shows the difference between single and multi-task learning settings. Multi-task learning is used to extract and share useful information across tasks to learn multiple tasks [23], [24]. With regard to the mechanism for the learning effect, multi-task learning provides five advantages to leveraging the prediction accuracy: implicit data augmentation to resolve a shortage of training data; attention focusing to achieve robustness against noisy data; eavesdropping, such as hints; representation bias for generalization; and regularization by introducing an inductive bias [25], [26]. Multi-task learning has shown an outstanding performance in various machine learning tasks, such as speech and face recognition, language processing, disease diagnosis, and quality prediction [27]–[31].

Moreover, beyond the merits of multi-task learning in the training phase, the number of predictive models can be significantly reduced. Because maintaining a high level of accuracy for the models of whole areas is labor-intensive, the use of multi-task learning improves the practicability of the proposed predictive approach.

Here, to leverage the prediction accuracy of the transfer request, we have the training dataset $[X^t, Y^t]^{e_t}$, where e_t denotes the number of samples in the t^{th} area. In addition, $X^t \in \mathbb{R}^d$ is the d-dimensional independent variable at t, and $Y^t \in \mathbb{R}^1$ is the dependent variable. We denote the set of the independent variables by $\mathbb{X} = (X^1, \ldots, X^t)$. Let the learning parameter $W = \{w^1, \ldots, w^t\}$ for each area.

Equations (1) and (2) provide the probabilistic model for the single- and multi-task models, respectively:

$$p(Y|f^w(X)) = \mathcal{N}(f^w(X), \sigma^2), \tag{1}$$

where the probability model identifies the mapping function f^w from X to Y, and is defined as a Gaussian likelihood function with zero mean and noise σ. For the multi-task model, factorization follows the set of conditional probabilities of tasks shown in Eq. (2):

$$\begin{aligned} & p\left(Y^1, \ldots, Y^t | f^W(\mathbb{X})\right) \\ & = p\left(Y^1 | f^{w^1}(\mathbb{X})\right) \ldots p\left(Y^t | f^{w^t}(\mathbb{X})\right) \\ & = \prod_{i=1}^t p\left(Y^i | f^{w^i}(\mathbb{X})\right) \prod_{i=1}^t \mathcal{N}\left(f^{w^i}(\mathbb{X}), (\sigma^i)^2\right), \end{aligned} \tag{2}$$

where i is the index of the tasks. Equation (2) denotes that the multi-task model converges to achieve high accuracy in a balanced manner for all tasks f^W. Regarding the maximum likelihood inference, the single- and multi-tasks have the log likelihoods shown in Equations. (3) and (4), respectively:

$$\log \mathcal{N}\left(f^w(X), \sigma^2\right) \propto -\frac{1}{2\sigma^2} \left\| Y - f^w(X) \right\|^2 - \log \sigma, \tag{3}$$

$$\log \prod_{i=1}^t \mathcal{N}\left(f^{w^i}(\mathbb{X}), (\sigma^i)^2\right) \propto \sum_{i=1}^t -\frac{1}{2(\sigma^i)^2} \left\| Y^i - f^{w^i}(\mathbb{X}) \right\|^2 - \log(\sigma^1 \ldots \sigma^i). \tag{4}$$

We state the minimization of a loss function $\mathcal{L}$ with the learning parameters W and σ^i as follows:

$$\begin{aligned} & \underset{W}{\operatorname{argmin}} \sum_{i=1}^t \mathcal{L}\left(W, \sigma^i\right) \\ & = -\log \prod_{i=1}^t \mathcal{N}\left(f^w(\mathbb{X}), \left(\sigma^i\right)^2\right) \\ & \propto \sum_{i=1}^t \frac{1}{2(\sigma^i)^2} \mathcal{L}^i(W) + \log(\sigma^1 \ldots \sigma^t), \end{aligned} \tag{5}$$

where $\mathcal{L}^i(W) = \left\| Y^i - f^{w^i}(\mathbb{X}) \right\|^2$. In Eq. (5), we optimize the parameter W to minimize the sum of the loss function $\mathcal{L}$, such as the mean square error (MSE), $\mathcal{L}\left(Y, \hat{Y}\right) = \left| Y - \hat{Y} \right|^2$.

According to Eq. (5), we have two implications. First, we obtain a more regularized model in identifying the optimal W^* that minimizes $\mathcal{L}$ for all tasks f^W. Further, multi-task learning forces faster and better convergence by reducing the search space of combinations of parameters W. This is due to finding a solution on a smaller space of representations by sharing useful information across tasks. Second, a small value of σ increases the contribution of $\mathcal{L}$ for the total losses, whereas a large value of σ decreases the contribution of $\mathcal{L}$. Thus, the noise σ^i can be seen as a weight scalar for the i^{th} task loss $\mathcal{L}^i$. Note that multi-task learning may have negative effects on the balancing act between bias and variance. There is a risk of underfitting to noisier tasks having a high σ^i in an attempt to minimize the total error, whereas overfitting less noisy tasks having a low σ^i [32].

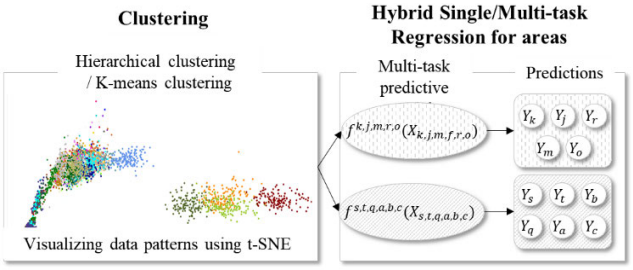


FIGURE 3. Proposed clustered multi-task learning for vehicle repositioning. To determine areas that have an unload transfer delay, we first partitioned the areas into several groups, and then constructed the multi-task models for each group.

In general, multi-task learning outperforms single-task learning under the assumption that tasks are related. Because multi-task learning reduces the sum of the loss function $\mathcal{L}$ of all the tasks, the generalization performance of multi-task models is better than that of the single-task models. However, if the tasks are unrelated, or the features extracted from each task may not help, we confront a negative transfer, which degenerates the prediction accuracy. Thus, it is hard to guarantee an expected performance of multi-task models [33]. To achieve the advantages of using multi-task learning without a negative transfer, it is necessary to apply multi-task learning for tasks if their independent or dependent variables have similar patterns. To resolve this, we adopt clustered multi-task learning, assuming that multiple tasks are partitioned into a set of clusters, as shown in Fig. 3. We use a hard parameter sharing structure for simpler and more explicit multi-task learning:

B. CLUSTERING FOR MULTI-TASK LEARNING

We established clustered multi-task learning by integrating unsupervised and supervised approaches sequentially to achieve the benefits of multi-task models. A clustering analysis partitions samples by minimizing within-group variations while maximizing between-group variations and then labels each sample with a cluster group [34].

Most related studies that combine clustering and multi-task learning have focused on a shared latent subspace, where relations between multiple tasks are more effectively learned [35]–[37]. Murugesan *et al.* [38] proposed incorporating co-clustering and multitask learning to learn a shared representation with task and feature clusters. Some studies have proposed an optimization method for multi-task learning, in that multi-task learning serves as a regularization term to obtain model smoothness [28], [32], [39]. However, these studies have overlooked the difficulties of interpreting how clustering is reflected in the training phase. When adopting a predictive approach in a real plant, explanations/interpretations for predictions are necessary to confirm normal predictive operations without allowing zero malfunction. Wang *et al.* [40] presented two-phase clustered multitask learning; first, a cluster analysis was conducted in terms of the predictions, and multi-task models were then built based on ensemble techniques.

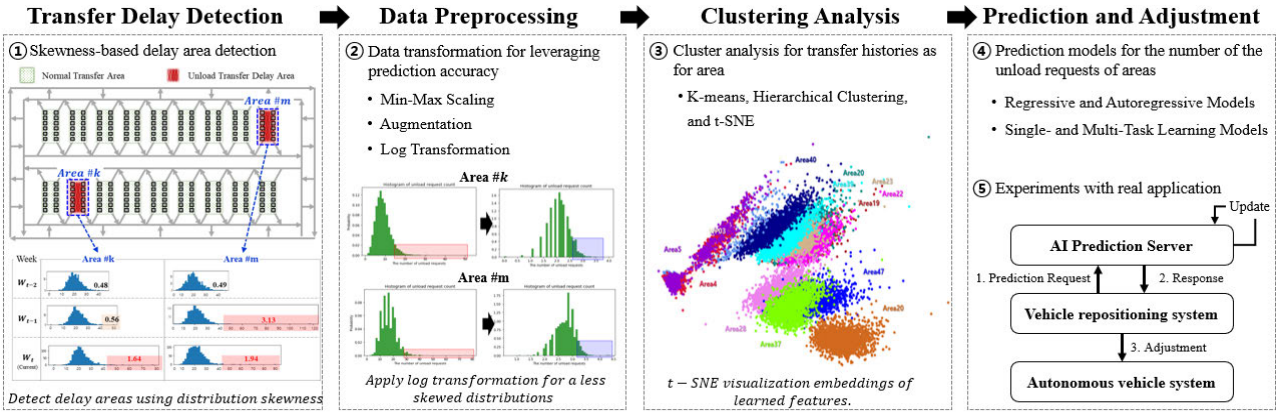


FIGURE 4. Procedure of the proposed approach with clustered multi-task learning.

IV. METHOD

A. OVERVIEW

Fig. 4 shows the procedure of the proposed approach. We gathered the data representing the traffic conditions and transfer requests for the areas. To achieve a better prediction accuracy, the datasets were transformed through min-max scaling, augmentation, and log transformations. Clustering analysis was conducted to determine the underlying heterogeneity within the areas. Finally, based on the clustering results, we built a multi-task seq2seq learning model to predict the number of transfer requests in areas. We compared two different settings: comparisons of the single- and multi-task learning models, and comparisons of the regressive and autoregressive models. Finally, the vehicle repositioning system used these predictions to adjust the idle vehicle repositioning control parameters in AVSs. Four steps are described in detail in Section IV-B to IV-H.

B. DATA DESCRIPTION

For the datasets, four months of datasets were gathered from a real semiconductor plant. The number of observations was approximately 11,500. The datasets include the number of transfer requests, the average and deviation of the delivery, waiting, unload, and load transfer times. The waiting time refers to the time duration from the transfer request to the vehicle assignment, whereas the load transfer time is the time from the pickup point to the destination. In addition, the average and deviation of the flow count, speed, congestion, and the number of unloading, loading, moving, and idle vehicles are included. We use these datasets to achieve two goals: 1) to identify the unload transfer delay areas and 2) to build the predictive area models.

First, to identify the areas where the unload transfer time is abnormally long, we use the Galton skewness coefficient (GSC), which estimates the degree of skewness of the distribution by $\frac{Q_1+Q_3-2Q_2}{Q_3-Q_1}$. The higher the GSC, the more severe the unload transfer time delay. Fig. 5 shows the abnormal unload transfer delayed areas when their GSCs increase over

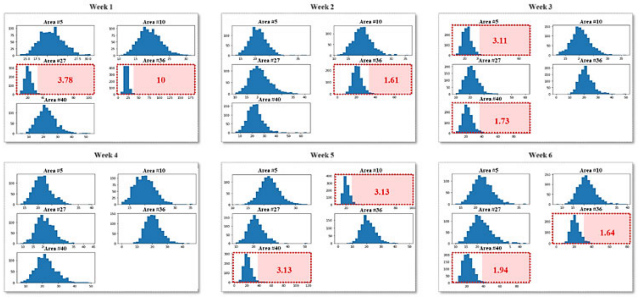


FIGURE 5. Histograms of unload transfer times measured at five sections for 6 weeks. The positive GSC indicates the tail on the right, meaning that there have been frequent unload transfer delays.

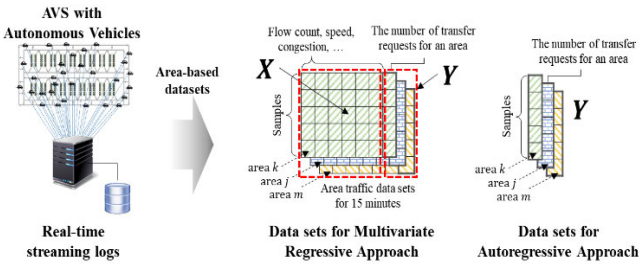


FIGURE 6. Data preparation and data structure for predictive models.

the predefined threshold value $\Theta_w = 1.6$ for the distribution of area w. The cases marked with the red squares indicate that unload transfer delays for areas frequently occur in terms of GSC. Indicating these delayed areas is essential to controlling the vehicle repositioning because we aim to minimize transfer delays in entire areas by properly distributing idle vehicles.

Second, we generate two different datasets based on the online data stream. Fig. 6 shows the overall procedure from real-time streaming data to the datasets to train the predictive models. The first datasets contain multivariate data for all areas to use regressive approaches. The second datasets contain a series of transfer requests for an area to use autoregressive approaches.

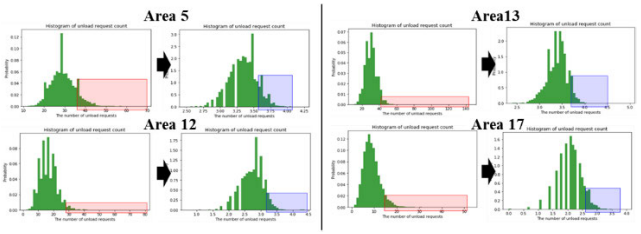


FIGURE 7. Log transformation for skewed distribution of four variables in areas 5, 12, 13, and 17.

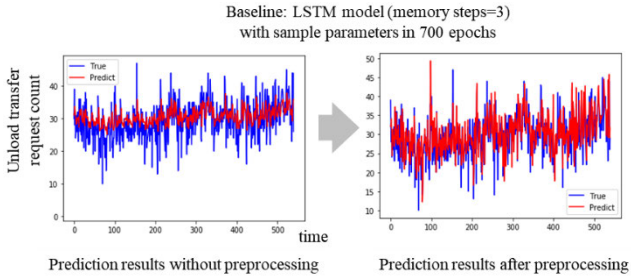


FIGURE 8. The effect of preprocessing with a log transformation and scaling in terms of the prediction results. This shows that the predicted values from preprocessing (red) are better fitted to the observed values (blue). Herein, LSTM is used with a time step of 3 as the baseline predictive model, and two results are compared after 700 epochs to show both the level of fit and the convergence speed.

C. PREPROCESSING

Regarding the preprocessing of the gathered dataset, we first convert the streaming transaction logs into area-based datasets. This is because we focus on determining the transport requests for each area. Second, for each area, we generate independent and dependent variables. As for the independent variables, we apply a log transformation to make a less skewed distribution of each variable. This is because a highly skewed distribution of a variable may cause an underfit for a sparse region having a few samples. To resolve the skewness problems, a log transformation is usually used [41]. By adopting a log transformation in the training phase as $y' = \ln(1 + y)$, the sparse region of the learning space within the entire distribution can be reduced by preprocessing the log transformation.

Fig. 7 illustrates that less skewed distributions can be obtained from the log transformation for the unload request counts of the four areas. We then adopt min-max scaling (using the minimal and maximal retention time) and adjusted features on a scale from (0 to 1). Min-max scaling realizes equal scaling of each variable. Finally, we verified the accuracy of the given datasets to ensure consistency of the data. These preprocessing steps significantly improve the prediction accuracy and convergence speed of the predictive models shown in Fig. 8.

D. PREDICTIVE MODEL

This section presents a model for predicting the number of transfer requests. Based on the predicted transfer requests for each area, we developed a vehicle repositioning system to autonomously distribute idle vehicle positions from excessively large areas to insufficient areas. However, inaccurate predictions cause inefficient vehicle repositioning and local traffic congestion, and increase the production starvation, resulting in production losses. To maximize the effectiveness of the proposed approach and minimize side effects, a highly accurate prediction is required. To achieve this, we propose using four variant models, as shown in Fig. 9: i) single-task multivariate regressive model, ii) multi-task multivariate regressive model, iii) single-task univariate autoregressive model, and iv) multi-task multivariate autoregressive model. We first evaluate the predictive performance of the regression models and time-series models. We then compare the single- and multi-task learning models.

The first and second approaches consider multivariate regressive models. Independent variables X contain 27 traffic conditions and transfer requests for areas, whereas the dependent variable Y is the number of transfer requests for the areas. During the training phase, we built a multivariate regressive model f_θ^k for area k as $f_\theta^k\left(X_{t-1}^k\right) = Y_t^k$, where θ denotes the model parameters. In addition, X_{t-1}^k are mapped to Y_t^k to predict the transfer requests after one time unit in an area of interest. The first approach is composed of independent models for each area. The second approach uses multi-task learning as $f^{k,j,m}\left(X_{t-1}^{k,j,m}\right) = Y_t^{k,j,m}$, where k, j, and m are related areas that are adjacent in terms of location, or where similar groups of facilities are located.

The third and fourth approaches consider autoregressive models. We first build a prediction model of each area independently for multi-step time-series forecasting, as $f^k\left(Y_{t-3,t-2,t-1}^k\right) = Y_t^k f^k\left(Y_{t-3,t-2,t-1}^k\right) = Y_t^k$. The number of time steps is empirically set to three. In the fourth approach, we propose using the multi-task multivariate autoregressive approach as $f^{k,j,m}\left(Y_{t-3,t-2,t-1}^{k,j,m}\right) = Y_t^{k,j,m}$, where areas k, j, and m are related. Here, we use a deep time-series model as a baseline. Section IV-6 provides all details of the variants of time-series forecasting for the third and fourth models.

E. TIME SERIES PREDICTION

For time series modeling, Fig. 10 shows that variants of the deep learning techniques are used for the third and fourth models. Overall, four deep learning techniques are considered: recurrent neural networks (RNNs), long short-term memory (LSTM), a gated recurrent unit (GRU), and Seq2Seq [42], [43]. An RNN has a decoder and an encoder network, and an RNN unit is usually one with a gating mechanism, such as an LSTM or a GRU. Seq2Seq also has an encoder–decoder network with a state vector, where a sequence of tokens is encoded into vectors by a recurrent network structure [44].

Fig. 10 shows the architecture of the RNN and Seq2Seq models. For recurrent unit cells, we can select an RNN, an LSTM, or a GRU. We set both the number of units in the input layer (the number of memory time steps in the

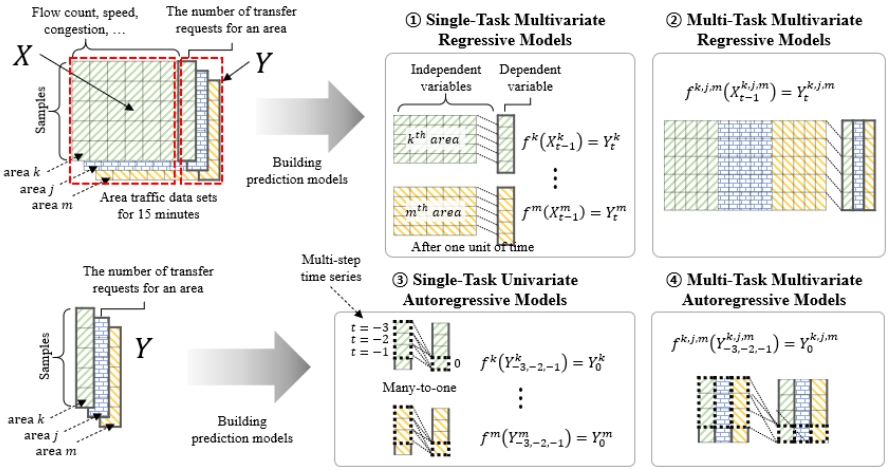


FIGURE 9. Comparative studies for predicting the number of unload transfer requests with four models.

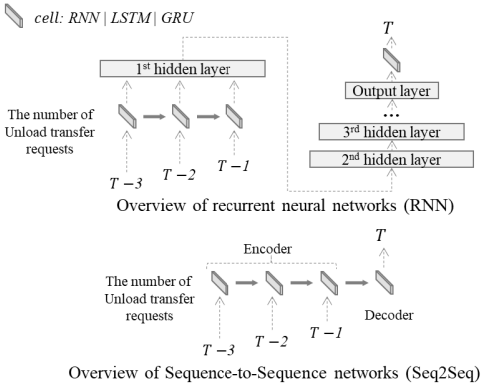


FIGURE 10. Schematic representation of the RNN and Seq2Seq models.

encoder), and the number of units in the output layer (the number of forecasting time steps in the decoder), to three. An RNN consists of one input layer, twenty hidden layers, and one output layer. Here, Seq2Seq consists of two recurrent networks, an encoder and a decoder, to extract a significant feature vector sequence. We use an empirical loss function to train the predictive model to determine the parameters, such as the network connection weights and vectors of the models.

In addition, we use the Nesterov-accelerated adaptive moment estimation (Nadam) optimizer to prevent overfitting, which guarantees a faster convergence than other gradient descent optimizers. We empirically set the hyperparameters and comparative algorithms as in Tables 1 and 2, respectively:

Further, regarding the number of dependent variables in prediction, we built single- and multi-task learning models to evaluate the effectiveness by sharing useful information between related areas. We thus present four types of models for comparisons using the regressive and autoregressive models and the single- and multi-task models. Further, regarding the multi-task models, we used clustering techniques to identify the closely related tasks. Fig. 9 summarizes the given data structure and four comparative models.

TABLE 1. Hyperparameter settings of the RNN and Seq2Seq models.

Parameter	Value
Epoch	5,000
Batch size	60
The ratio of training and testing sets	Ratios of 0.8 and 0.2
The number of memory time steps	3
The number of forecasting time steps	3
The number of cell units	160
The number of dense units (RNN)	20
Optimizer	Nesterov-accelerated adaptive moment estimation (Nadam)
Learning rate	0.0002
β_1	0.9
β_2	0.999
Loss function	Mean squared error (MSE)
The number of epoch patience for model checkpoint and early stopping options	700

F. CLUSTERED MULTI-TASK LEARNING

Here, we adopt two-stage clustered multi-task learning to obtain both high prediction accuracy and interpretability for clustering results. Prior to cluster analysis, we use t-stochastic neighbor embedding (t-SNE) to visualize the intrinsic structure of the given dataset, and explore the embedding space based on the area in Fig. 11. Regarding a cluster analysis, we conduct hierarchical and k-means clustering methods. To obtain reliable clustering results, we compared the experimental results of the two methods with the domain expert knowledge. The hyperparameters are empirically set as follows: similarity measure = ward distance, distance threshold = infinite, and silhouette index is used for selecting the best number of clusters k^* for k-means, and cutting the tree for hierarchical clustering. As the final clusters, we accept the intersections of the clusters obtained from the results of the two methods. The intersections were grouped into eight clusters of areas. We confirm that a

TABLE 2. Comparative algorithms. These algorithms are used for both the single- and multi-task approaches.

Category	Parameter	Value	Task	Model
Regressive approach	Plain neural networks	12 hidden layers activation function = Leaky ReLU batch normalization and dropout used	Single	Single-task multivariate regressive models
			Multiple	Multi-task multivariate regressive models
Univariate/ Multivariate Auto-regressive approach	Recurrent neural networks	Model = RNN, cell = plain RNN	Single	Single-task univariate autoregressive models
	Long shot term memory	Model = RNN, cell = LSTM		
	RNN with gated recurrent unit	Model = RNN, cell = GRU		
	Sequence to sequence model with LSTM	Model = Seq2Seq, cell = LSTM	Multiple	Multi-task multivariate autoregressive models
	Sequence to sequence model with GRU	Model = Seq2Seq, cell = GRU		

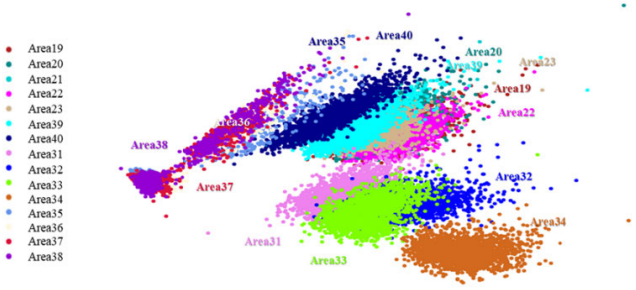


FIGURE 11. *Visualization with t-Stochastic neighbor embedding for different areas.*

multi-task model is constructed based on the clustering results. Fig. 12 shows the clustering exemplars of k-means with a PCA-based dimensionality reduction under the experimental tests for the number of clusters k.

G. PERFORMANCE EVALUATION

To evaluate the predictive performance, the dataset is divided into training and test sets with a ratio of 8 to 2. We use the mean absolute error (MAE) and root mean square error (RMSE) measures for an accuracy performance. The MAE and RMSE are calculated as follows:

$$\text{Mean absolute error (MAE)} = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right|, \quad (6)$$

$$\text{Root mean square error (RMSE)} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right|^2}, \quad (7)$$

where y_j and $\hat{y}_j$ indicate the j^{th} actual and predicted values, respectively, and n denotes the sample size. The RMSE penalizes a high weight to large errors $(y_i - \hat{y}_i)$ because the squared errors are averaged. Meanwhile, the MAE is easy to understand, and is a useful statistic for comparing the predictive performance under a quadratic loss.

H. PREDICTION RESULTS

Table 3 presents the predicted results for six significant areas in terms of the number of unload requests with the four comparative models described in Section IV-C. Overall, the proposed clustered multi-task sequence-to-sequence learning (CMSL), in which the dashed border-box indicates, shows outperformance in most areas. Specifically, we observe that the multi-task approach outperforms the single-task approach in all areas. A comparison of the regressive and autoregressive approaches shows that the autoregressive model achieves relatively high accuracy. Fig. 13 shows that the proposed CMSL predictions for the number of unload requests are properly applied.

V. FIELD APPLICATION

A. SYSTEM CONFIGURATION

Simulations were conducted to evaluate the applicability of the proposed method regarding the transfer performance of AVSs. We used Python 3.7.1, and Java SE Development Kit 1.8.0_212 to implement the testbed platform. We also used the scikit-learn Python library and Keras with TensorFlow (an open-source library to provide low-level blocks for neural networks) to train the learning models. We developed an asynchronous HTTP server and client pair to manage a non-blocking prediction messaging service between the prediction system and the AVS simulator for real-time simulation. Simulations were run on a PC with Windows 10, a 3.40 GHz i7-6700 CPU, and 32 GB of RAM.

We built a test platform for a real-time simulation that can address the dynamic vehicle routing problem with the concept drift phenomenon. Fig. 14 shows the specifications of the proposed system containing the proposed prediction system and AVS simulation testbed. The prediction system contains four modules: prediction manager, model repository, model adaptation, and change detector. The AVS testbed contains five major modules: a traffic controller, core controller, data collector, vehicle emulator, and simulation manager.

A real-time simulation between systems is run using the following procedure: The data collector first gathers traffic data. The traffic controller requests a prediction of the traffic conditions in bottleneck sections once every 15 min. The prediction manager receives the request and executes models to predict the traffic for these sections. When the system returns the predicted traffic information, it is sequentially reflected in the simulator through the traffic controller, core controller, and vehicle emulator. In detail, the core controller adjusts the routing configuration parameters, which correspond to the penalty costs of the sections for a route calculation. Thousands of vehicle emulators are then assigned to the route with the updated link penalties. The model adaptation with the change detector is conducted by comparing the actual and predicted traffic data that can be used to update the model.

B. RESULTS

Tables 4 and 5 demonstrate the effectiveness of the proposed CMSL vehicle repositioning system in terms of the average

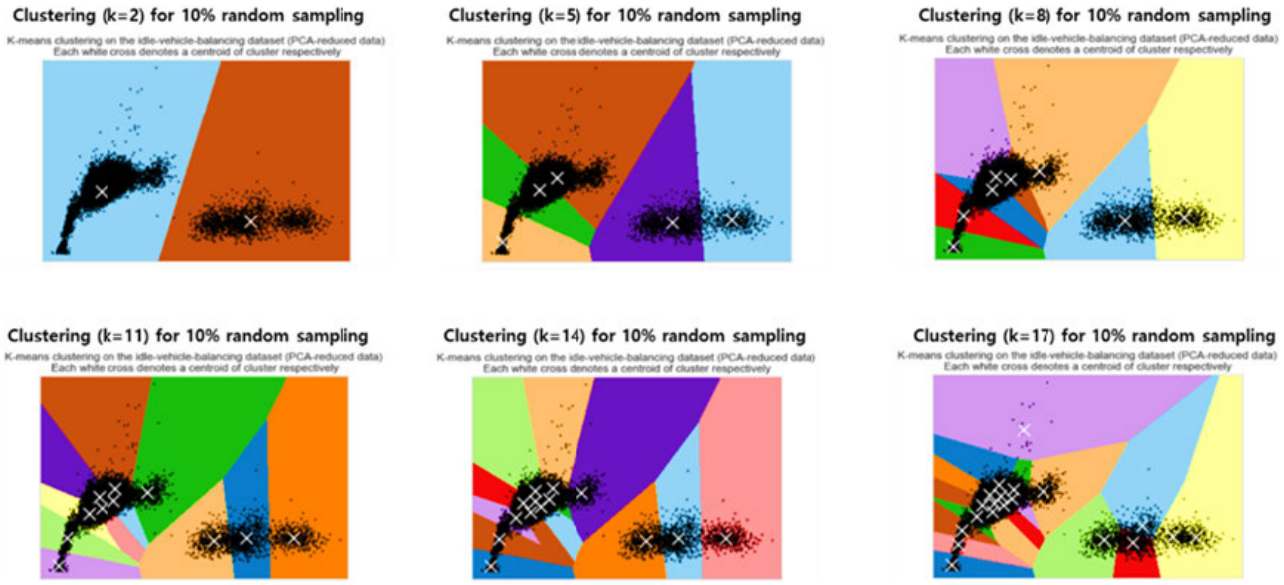


FIGURE 12. Clustering experiments in terms of the number of clusters k.

TABLE 3. Prediction results with single- and multi-task multivariate regressive models. The dashed border box shows the results of the proposed CMSL (Bold face denotes the lowest errors).

Area No.	Multivariate regressive model		Multivariate/Univariate autoregressive model									
			Single-task learning					Clustered multi-task learning				
	Single-task learning	Clustered multi-task learning	RNN	LSTM	RNN (GRU)	Seq2Seq (LSTM)	Seq2Seq (GRU)	RNN	LSTM	RNN (GRU)	Seq2Seq (LSTM)	Seq2Seq (GRU)
Area 19	0.1024	0.0962	0.0746	0.0955	0.0847	0.0828	0.0626	0.0859	0.0831	0.0639	**0.0197**	0.025
Area 20	0.158	0.123	0.0751	0.0955	0.0781	0.0729	0.0569	0.0729	0.041	0.0729	**0.0107**	0.0152
Area 22	0.0516	0.0508	0.1283	0.0971	0.0829	0.102	0.0943	0.0749	0.0579	0.0364	0.0142	**0.0121**
Area 23	0.0706	0.0666	0.0727	0.0945	0.0778	0.0786	0.0493	0.0831	0.0437	0.0418	**0.0115**	0.0132
Area 39	0.0834	0.0822	0.0732	0.0964	0.0926	0.068	0.0657	0.0926	0.045	0.0451	0.0138	**0.0131**
Area 40	0.026	0.019	0.0727	0.0951	0.0857	0.0681	0.0637	0.0857	0.044	0.051	**0.0126**	0.0163

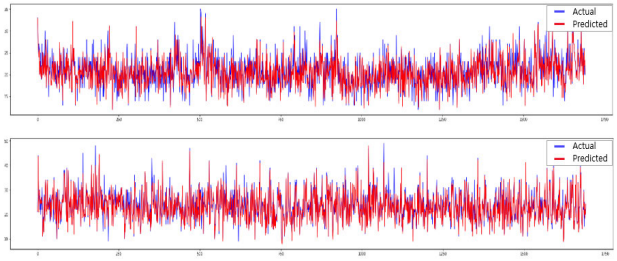


FIGURE 13. Prediction results for test datasets in terms of number of unload requests for the 32nd and 33rd areas; the x-axis is the time (in 15-min intervals), whereas the y-axis is the number of unload requests.

TABLE 4. Experimental results of the existing and proposed systems regarding the sum of waiting and unload request times.

Area ID	The existing system		The proposed system		P-values for one-sided Kolmogorov–Smirnov (KS) tests ($\alpha = 0.05$)
	Mean	Standard deviation	Mean	Standard deviation	
Area 19	44.721	0.372	**41.279**	0.439	**< 0.001***
Area 20	38.743	0.080	**37.182**	0.207	**< 0.001***
Area 22	45.583	0.367	**45.029**	0.266	**< 0.003***
Area 23	51.944	0.457	**45.284**	0.349	**< 0.001***
Area 39	37.703	0.064	**36.054**	0.451	**< 0.001***
Area 40	**38.465**	0.239	40.595	0.309	1
Total	53.600	0.134	**50.468**	0.168	**< 0.001***

unload request times and delays. Table 4 shows that the total unload transfer times in the proposed system were reduced by 5.84%. In addition, the performance of each area among the six key areas is better than that of the existing system, except for area 40. Table 5 shows that the sum of all numbers of unload transfer delays of over 120 s in the proposed system was reduced by 43.1%. Likewise, the unload transfer delay count of individual areas was improved. We conducted Kolmogorov–Smirnov (KS) tests to statistically validate whether the rank of the population means differed

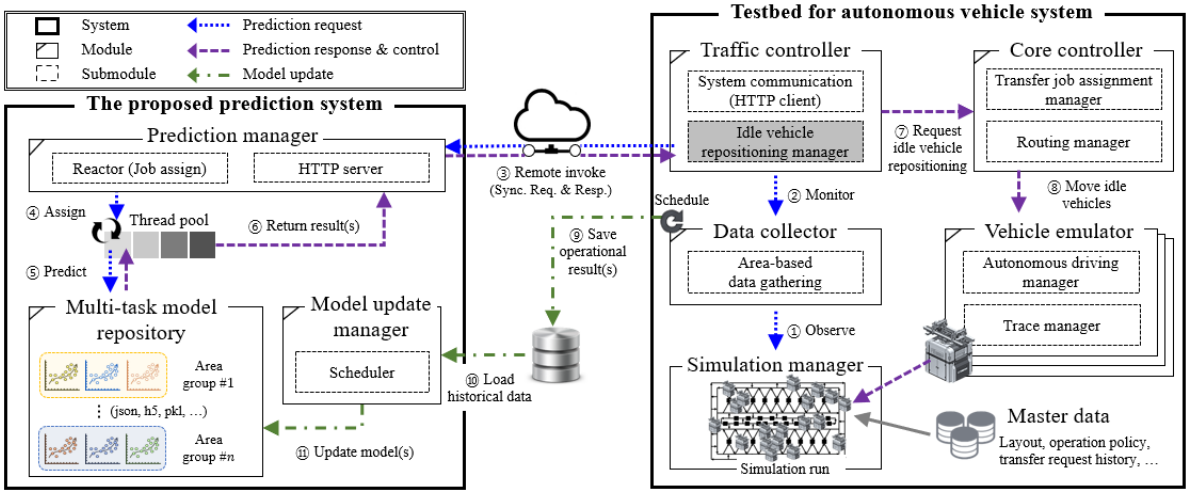


FIGURE 14. Proposed predictive framework for field application.

TABLE 5. Experimental results of the existing and proposed systems regarding the number of abnormally long-term delay transfers over 120 s.

Area ID	The existing system		The proposed system		*P*-values for one-sided Kolmogorov–Smirnov (KS) tests ($\alpha = 0.05$)
	Mean	Standard deviation	Mean	Standard deviation	
Area 19	20.4	0.400	**10.4**	< 0.000	**< 0.001***
Area 20	**1**	< 0.000	**0.6**	< 0.000	**< 0.001***
Area 22	35.8	0.748	**28**	1.166	**< 0.001***
Area 23	34.4	0.490	**23.2**	0.433	**< 0.001***
Area 39	2	0.400	2	< 0.000	0.407
Area 40	21.6	0.400	**16.6**	< 0.000	**< 0.001***
Total	801.4	4.118	**455.8**	5.953	**< 0.001***

significantly between the existing and proposed systems. The p-values observed for most areas were less than the significance level $\alpha = 0.05$, indicating that the performance difference between the existing and proposed systems was statistically significant.

These unload transfer times and delays have a significant impact on productivity. Owing to the reduction of the unload transfer time, the lead time required to complete the products can be reduced. In addition, a decrease in transfer delays can improve productivity because transfers may decrease the efficiency of the production machines, impeding the production schedules, resulting in production losses. The results demonstrate that the proposed CMSL-based system significantly reduces the number of long-term delay transfers. This is because the proposed method repositions idle vehicles based on predictive unload transfer requests. As intended, the results show that idle vehicles are adequately placed in an area that requests unload transfers.

VI. CONCLUDING REMARKS

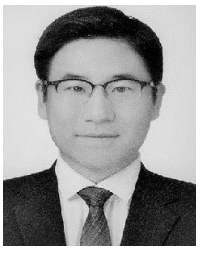
This study aims to model and implement a robust vehicle repositioning system using a predictive approach for autonomous vehicle systems in a large-scale semiconductor plant. In particular, we set the target to achieve a practical solution for autonomous vehicle systems in large-scale semiconductor plants. We gathered the streaming data describing dynamic manufacturing environments, and thus analyzed the areas where the unload transfer delay occurred because the idle vehicle was not properly placed. We proposed a predictive approach for idle vehicle repositioning to minimize unload transfer delays and maximize machine utilization.

For the predictive models, we proposed CMSL models to obtain a better prediction accuracy with multiple areas by considering autoregressive patterns in related areas. We explicitly clustered similar area groups by using a clustering analysis, and then modeled multi-task models to leverage the prediction accuracy. The proposed model was compared to state-of-the-art competitive algorithms and models. Experimental results from the field application showed that the proposed system reduced the average unload transfer times and decreased the number of unload transfer delays. This means that our proposed model can achieve a crucial improvement in productivity.

In future studies, we plan to extend the current studies to control the weight for the significant levels of areas, and build idle vehicle control modules that are more advanced for more complex scenarios with high uncertainties, such as traffic congestion, transfer request cancels, and production increases or decreases. The proposed CMSL is expected to promote the trend of adopting a predictive approach by providing a high and robust accuracy performance under dynamic production conditions. In addition, lack of data is a chronic problem in real-world AMHSs. Thus, active learning would be useful to improve learning efficiency.

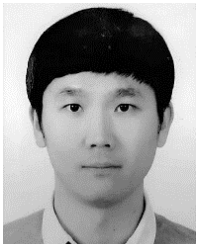
REFERENCES

[1] S. Lee, H.-G. Kahng, T. Cheong, and S. B. Kim, "Iterative two-stage hybrid algorithm for the vehicle lifter location problem in semiconductor manufacturing," *J. Manuf. Syst.*, vol. 51, pp. 106–119, Apr. 2019.
[2] S. Lee, H. J. Kim, and S. B. Kim, "Dynamic dispatching system using a deep denoising autoencoder for semiconductor manufacturing," *Appl. Soft Comput.*, vol. 86, Jan. 2020, Art. no. 105904.
[3] S. Lee, Y. Kim, H. Kahng, S. Lee, S. Chung, T. Cheong, K. Shin, J. Park, and S. B. Kim, "Intelligent traffic control for autonomous vehicle systems based on machine learning," *Expert Syst. Appl.*, vol. 144, Apr. 2020, Art. no. 113074.
[4] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning for dynamic time series prediction," *Appl. Soft Comput.*, vol. 70, pp. 576–589, Sep. 2018.
[5] P. J. Egbelu, "Positioning of automated guided vehicles in a loop layout to improve response time," *Eur. J. Oper. Res.*, vol. 71, pp. 32–44, Nov. 1993.
[6] K. H. Kim, "Positioning of automated guided vehicles in a loop layout to minimize the mean vehicle response time," *Int. J. Prod. Econ.*, vol. 39, no. 3, pp. 201–214, May 1995.
[7] A. J. R. M. Gademann and S. L. van de Velde, "Positioning automated guided vehicles in a loop layout," *Eur. J. Oper. Res.*, vol. 127, no. 3, pp. 565–573, Dec. 2000.
[8] C. Lee and J. A. Ventura, "Optimal dwell point location of automated guided vehicles to minimize mean response time in a loop layout," *Int. J. Prod. Res.*, vol. 39, no. 17, pp. 4013–4031, Jan. 2001.
[9] S. Lee, "Locating idle vehicles in tandem-loop automated guided vehicle systems to minimize the maximum response time," *IEMS*, vol. 6, no. 2, pp. 125–135, 2007.
[10] C.-J. Huang, K.-H. Chang, and J. T. Lin, "Optimal vehicle allocation for an automated materials handling system using simulation optimisation," *Int. J. Prod. Res.*, vol. 50, no. 20, pp. 5734–5746, Oct. 2012.
[11] K.-H. Chang, Y.-H. Huang, and S.-P. Yang, "Vehicle fleet sizing for automated material handling systems to minimize cost subject to time constraints," *IIE Trans.*, vol. 46, no. 3, pp. 301–312, Mar. 2014.
[12] Y.-K. Lin and P.-C. Chang, "A novel reliability evaluation technique for stochastic-flow manufacturing networks with multiple production lines," *IEEE Trans. Rel.*, vol. 62, no. 1, pp. 92–104, Mar. 2013.
[13] J. T. Lin and C.-J. Huang, "A simulation-based optimization approach for a semiconductor photobay with automated material handling system," *Simul. Model. Pract. Theory*, vol. 46, pp. 76–100, Aug. 2014.
[14] J. T. Lin, F. K. Wang, and C. J. Yang, "The performance of the number of vehicles in a dynamic connecting transport AMHS," *Int. J. Prod. Res.*, vol. 43, no. 11, pp. 2263–2276, Jun. 2005.
[15] B. Kim and J. Park, "Idle vehicle circulation policies in a semiconductor FAB," *J. Intell. Manuf.*, vol. 20, p. 709, Aug. 2009.
[16] B. Vahdani, "Vehicle positioning in cell manufacturing systems via robust optimization," *Appl. Soft Comput.*, vol. 24, pp. 78–85, Nov. 2014.
[17] J. T. Lin, C.-H. Wu, and C.-W. Huang, "Dynamic vehicle allocation control for automated material handling system in semiconductor manufacturing," *Comput. Oper. Res.*, vol. 40, no. 10, pp. 2329–2339, Oct. 2013.
[18] J. E. Kiba, S. Dauzère-Pérès, C. Yugma, and G. Lamiable, "Comparing transport policies in a full-scale 300mm wafer manufacturing facility," in *Proc. 11th IMHRC*, Milwaukee, WI, USA, 2010, pp. 1–15.
[19] J. A. Jimenez, A. Grosser, C. Adikaram, V. Davila, R. Wright, and M. Bell, "AMHS factors enabling small wafer lot manufacturing in semiconductor wafer FABS," in *Proc. Winter Simulation Conf.*, Dec. 2010, pp. 2575–2585.
[20] A. Ben Chaabane, S. Dauzere-Peres, C. Yugma, L. Rulliere, and G. Lamiable, "Analyzing the impact of key parameters of vehicle management policies in a unified AMHS," in *Proc. Winter Simulations Conf. (WSC)*, Dec. 2013, pp. 3818–3828.
[21] R. Schmaler, T. Schmidt, M. Schoeps, J. Luebke, R. Hupfer, and N. Schlaus, "Simulation based evaluation of different empty vehicle management strategies with considering future transport jobs," in *Proc. Winter Simulation Conf. (WSC)*, Dec. 2017, pp. 3576–3587.
[22] K. Ahn and J. Park, "Idle vehicle rebalancing in semiconductor fabrication using factorized graph neural network reinforcement learning," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 132–138.
[23] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *J. Mach. Learn. Res.*, vol. 6, pp. 1817–1853, Nov. 2005.
[24] C. Park, Y. Kim, Y. Park, and S. B. Kim, "Multitask learning for virtual metrology in semiconductor manufacturing systems," *Comput. Ind. Eng.*, vol. 123, pp. 209–219, Sep. 2018.
[25] J. Baxter, "A model of inductive bias learning," *J. Artif. Intell. Res.*, vol. 12, pp. 149–198, Mar. 2000.
[26] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, *arXiv:1706.05098*. [Online]. Available: http://arxiv.org/abs/1706.05098
[27] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 160–167.
[28] T. T. Allen, S. Roychowdhury, and E. Liu, "Reward-based Monte Carlo-Bayesian reinforcement learning for cyber preventive maintenance," *Comput. Ind. Eng.*, vol. 126, pp. 578–594, Dec. 2018.
[29] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 4835–4839.
[30] X. Yin and X. Liu, "Multi-task convolutional neural network for pose-invariant face recognition," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 964–975, Feb. 2018.
[31] Y. Hong, B. Wei, Z. Han, X. Li, Y. Zheng, and S. Li, "MMCL-Net: Spinal disease diagnosis in global mode using progressive multi-task joint learning," *Neurocomputing*, vol. 399, pp. 307–316, Jul. 2020.
[32] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7482–7491.
[33] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017, *arXiv:1707.08114*. [Online]. Available: http://arxiv.org/abs/1707.08114
[34] R. Zhang and A. I. Rudnicky, "A large scale clustering scheme for kernel K-means," in *Proc. Object Recognit. Supported User Interact. Service Robots*, Aug. 2002, pp. 289–292.
[35] J. Zhou, J. Chen, and J. Ye, "Clustered multi-task learning via alternating structure optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 702–710.
[36] M. Li, C. Liu, K. Li, X. Liao, and K. Li, "Multi-task allocation with an optimized quantum particle swarm method," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106603.
[37] W. Yang, Z. Li, C. Wang, and J. Li, "A multi-task faster R-CNN method for 3D vehicle detection based on a single image," *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106533.
[38] K. Murugesan, J. Carbonell, and Y. Yang, "Co-clustering for multi-task learning," 2017, *arXiv:1703.00994*. [Online]. Available: http://arxiv.org/abs/1703.00994
[39] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 527–538.
[40] M. Wang, X. Zheng, and K. Zhang, "User-sensitive recommendation ensemble with clustered multi-task learning," 2018, *arXiv:1804.10795*. [Online]. Available: http://arxiv.org/abs/1804.10795
[41] H. Son, C. Hyun, D. Phan, and H. J. Hwang, "Data analytic approach for bankruptcy prediction," *Expert Syst. Appl.*, vol. 138, Dec. 2019, Art. no. 112816.
[42] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*. [Online]. Available: http://arxiv.org/abs/1409.1259
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

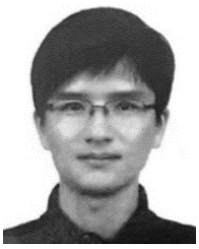


SANGMIN LEE received the M.S. degree in industrial engineering from KAIST, in 2007, and the Ph.D. degree in industrial engineering from Korea University, in 2019. He is currently a Professor with the School of Information Convergence, Kwangwoon University. His research interests include artificial intelligent techniques for various problems appearing in science and engineering. He has expertise in the novelty detection and concept drift learning.

DAE-EUN LIM received the B.S. degree in industrial engineering from Korea University, Republic of Korea, in 2004, and the M.S. and Ph.D. degrees in industrial engineering from KAIST, in 2006 and 2009, respectively. He is currently an Associate Professor with the Department of Industrial Engineering, Kangwon National University. His research interests include stochastic modeling and its applications to manufacturing systems.



YOUNKOOK KANG received the B.S. degree in industrial engineering from Korea University, in 2009, and the M.S. degree in industrial engineering from Seoul National University, Seoul, South Korea, in 2019. He worked as a Software Engineer with Samsung Electronics. He has expertise in predictive analytics and reinforcement learning for manufacturing.



HAE JOONG KIM received the B.S., M.S., and Ph.D. degrees in industrial engineering from Seoul National University, in 2001, 2003, and 2008, respectively. He has been working as a Software Engineer at Samsung Electronics, since 2008. His research interests include mathematical optimization and machine learning approaches for manufacturing systems.