Uncertainty-Aware Transfer Learning to Evolve Digital Twins for Industrial Elevators

Qinghua Xu
Simula Research Laboratory and University of Oslo
Oslo, Norway
qinghua@simula.no

Shaukat Ali
Simula Research Laboratory
Oslo, Norway
shaukat@simula.no

Tao Yue
Simula Research Laboratory
Oslo, Norway
tao@simula.no

Maite Arratibel
Orona
San Sebastian, Spain
marratibel@orona-group.com

ABSTRACT

Digital twins are increasingly developed to support the development, operation, and maintenance of cyber-physical systems such as industrial elevators. However, industrial elevators continuously evolve due to changes in physical installations, introducing new software features, updating existing ones, and making changes due to regulations (e.g., enforcing restricted elevator capacity due to COVID-19), etc. Thus, digital twin functionalities (often built on neural network-based models) need to evolve themselves constantly to be synchronized with the industrial elevators. Such an evolution is preferred to be automated, as manual evolution is time-consuming and error-prone. Moreover, collecting sufficient data to re-train neural network models of digital twins could be expensive or even infeasible. To this end, we propose unceRtaInty-aware tranSfer lEarning enriched Digital Twins **(RISE-DT)**, a *transfer learning* based approach capable of transferring knowledge about the waiting time prediction capability of a digital twin of an industrial elevator across different scenarios. RISE-DT also leverages *uncertainty quantification* to further improve its effectiveness. To evaluate RISE-DT, we conducted experiments with 10 versions of an elevator dispatching software from Orona, Spain, which are deployed in a Software in the Loop (SiL) environment. Experiment results show that RISE-DT, on average, improves the Mean Squared Error by 13.131% and the utilization of uncertainty quantification further improves it by 2.71%.

CCS CONCEPTS

• **Software and its engineering → Empirical software validation**; **Software evolution**; **Maintaining software**; **Embedded software**.

KEYWORDS

Digital Twin, Transfer Learning, Uncertainty, Industrial Elevators

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11…$15.00
https://doi.org/10.1145/3540250.3558957

ACM Reference Format:
Qinghua Xu, Shaukat Ali, Tao Yue, and Maite Arratibel. 2022. Uncertainty-Aware Transfer Learning to Evolve Digital Twins for Industrial Elevators. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3540250.3558957

1 INTRODUCTION

Elevators are commonly used in our daily lives, especially in high-rise buildings. Consequently, their incorrect operations or poor performance affect user experiences to various extents and, in extreme cases, may compromise users' safety [40]. To ensure robust and safe operations of elevator systems, digital twin technologies (DT) are promising solutions [18]. El Saddik defined a DT as "a digital replica of a living or non-living physical entity" [21]. Several researchers extend this concept and equip DTs with neural network models [60, 63]. However, elevator systems are prone to changes continuously due to, for instance, the variability in physical installation conditions, software updates and configurations caused by regulations, performance enhancements, etc [47, 58]. Therefore, to maintain its evolution – one of the key characteristics of DT technologies, a DT has to evolve its own functionalities to be able to continuously keep synchronized with its counterpart, i.e., the industrial elevator [64]. Such an evolution is expected to be automated, because it is error-prone and time-consuming (thus expensive) to manually update the DT.

Moreover, it is well-known that neural networks require a large amount of labeled training data to be effective; however, it is often expensive (if feasible) to collect sufficient data to re-train neural network models of the DT due to its evolution. Thus, there is an indispensable need for transferring knowledge gained in one operating scenario of an elevator to another, for important DT functionalities built with neural network models. To this end, we propose **RISE-DT**, a novel approach that utilizes transfer learning for evolving the passengers' waiting time prediction capability of DTs of industrial elevators across various elevator operating scenarios. Our industrial partner is Orona[1] – a Spanish company that manufactures elevators for different types of building configurations and setups. Data required for transfer learning were systematically collected via a Software in the Loop (SiL) simulation setting of an

[1] https://www.orona-group.com/

industrial elevator dispatching software from Orona and a commercial software–Elevate[2]. Elevate has been successfully applied in the literature for supporting validations of elevator systems [3, 4].

To further improve the performance of transfer learning, we also employ uncertainty quantification (UQ), which was initially designed to evaluate the robustness of machine learning models [52]. Current UQ methods (e.g., Bayesian methods [39] or ensemble methods [45]) mostly quantify the uncertainty of either a dataset or a model itself. Researchers have also developed several open source UQ tools, such as Uncertainty Wizard [61] and Uncertainty Toolbox [12]. Especially, Uncertainty Toolbox is a Pytorch-based tool, which can be easily integrated into our Pytorch-based implementation of RISE-DT. We therefore employ *Uncertainty Toolbox* to calculate the calibration and sharpness metrics for the purpose of selecting the most uncertain samples from the source dataset. Calibration demonstrates the consistency between the prediction distribution and the observation, while sharpness shows the concentration of the prediction distribution [25]. However, directly picking these samples from the source dataset leads to the missing of context information. Therefore, we feed selected uncertain samples into a multi-head attention module [57], aiming at preserving context information in new sample vectors.

As previously said, we evaluated RISE-DT with 10 versions of an industrial dispatching software from Orona in a SiL setting enabled with Elevate, which simulates elevator hardware, buildings, etc. We performed experiments on four different elevator scenarios and 10 dispatchers. Results show that, on average, RISE-DT improves Mean Squared Error (MSE) by 13.131% with transfer learning and the UQ further improves its performance by 2.71%.

Our key contributions are that: 1) We propose a novel model, RISE-DT, which takes advantage of both neural networks and DTs. Neural networks help us to learn complex patterns in elevator data, while DTs simulate the elevator in real-time. Combining neural networks and DTs allows RISE-DT to make accurate predictions in unforeseen situations, e.g., long waiting times for elevator passengers; 2) We mitigate the data scarcity problem of neural networks by introducing transfer learning. Transfer learning transfers knowledge across different elevator scenarios, which differs in dispatching algorithms or traffic templates; and 3) We employ the Uncertainty Toolbox [12] to perform UQ for each data sample (i.e., properties for passengers such as destination and mass), and a multi-head attention mechanism for preserving context information, i.e., information of previous passengers.

We present the industrial context in Section 2. Section 3 introduces RISE-DT, Section 4 describes experiment design, and Section 5 presents results. We present the related work in Section 6 and conclude the paper in Section 7.

2 INDUSTRIAL CONTEXT

Orona– a Spanish company, develops various types of vertical transportation systems for buildings. Each building is typically installed with a set of elevators. A dedicated controller controls the movements of an elevator. All the controllers are linked to a component called *traffic master* with a dedicated software component called *dispatcher* deployed, which is responsible for optimal scheduling of the elevators by providing the best possible Quality of Service (QoS), e.g., the minimized average waiting time (AWT). AWT, for instance, tells how much, on average, passengers have to wait for an elevator in a given time period.

Elevator installations vary from one building to another. How the elevators are used, i.e., *traffic*, also varies based on various parameters such as the building type, time of the day, and day of the week. To be cost-effective in supporting new installations, new types of traffic of elevators, and new versions of dispatchers, DTs of the elevators should adapt to such new changes timely. Thus, it is needed to learn knowledge gained, e.g., from one installation to another installation, one traffic type to another, or one version of a dispatcher to another.

Collecting operating data from physical elevators is expensive as physical data collection devices need to be set up. So, we collect data in a Software in the Loop (SiL) setting with industrial dispatchers from Orona situated in the commercial simulator Elevate. Elevate allows you to create various operating scenarios via configuring building types, passengers, etc. A notable feature is *traffic template*, which simulates traffic of a specific elevator scenario. For example, the *Lunch Peak* template simulates traffic in an office building where passengers are taking elevators to go to the floor where the canteen is located.

In this paper, we focus on predicting waiting times of passengers. For convenience, we first define the concept of *Elevator Scenario*. An elevator scenario C has two parts: an elevator dispatcher and its traffic template. First, an elevator dispatcher is usually deployed, in practice, in a particular setting, such as a configured SiL or real operation with particular settings (e.g., building setup). The dispatcher schedules passenger calls of elevators, by ensuring optimal QoS (e.g., minimal waiting time), and deals with various traffic patterns specified as traffic templates for SiL. Such traffic templates correspond to the real passenger traffic in the real operation configuration. Data generated from any of these configurations can be used for the construction of a neural network-based DT, which consists of two components: DT model (DTM) and DT capability (DTC). DTM is a live simulation of the elevator scenario, while DTC is built with machine learning algorithms for specific tasks, i.e. predicting waiting time wt for passengers in this context.

We aim to address three main challenges. First, like in any other domain, the good performance of neural network models comes from training on sufficient labeled data, in our case, which is about elevator data with waiting time for each passenger. However, sufficient labeled data for certain elevator scenarios can be unavailable. For example, if we have a newly upgraded elevator dispatcher, it is impossible to get operational data before it is actually deployed in the real environment. Our solution to this challenge is to transfer knowledge from other elevator scenarios with data ready to use.

Second, manually transferring knowledge across elevator scenarios requires significant time and expertise. To do this transfer learning manually, one needs to distill common knowledge shared among scenarios, e.g., passenger properties that potentially lead to unexpected long waiting time. If this is successful, expertise is required to apply this knowledge in the target scenario, which is about building a new model for the target scenarios. In comparison, our solution is to develop a neural network-based transfer learning architecture that can accomplish this transfer automatically.

[2] https://peters-research.com/index.php/elevate/

Third, potential elevator scenarios that can be used as transfer learning sources are insufficient. To make full use of transfer learning, we expect source elevator scenarios to be abundant and preferably heterogeneous, so that the final model can generalize well. However, this cannot be guaranteed in our case, as we have limited elevator scenario data. Hence, RISE-DT uses UQ to select the most uncertain samples, whose Shannon entropy tends to be higher and thus contains more information [42], which motivates us to take full advantage of these uncertain samples. Thus, we perform UQ to select most uncertain samples. However, elevator scenario samples are interdependent given their time series nature, so selecting such samples without considering their dependencies could potentially lose context information. To overcome this problem, we employ a multi-head attention module to preserve context information. We use the output of attention module as new vectors for each sample. In short, we aim to improve the effectiveness of transfer learning with the help of UQ and multi-head attention.

3 APPROACH

The core of RISE-DT is transfer learning, which transfers knowledge from one elevator scenario to another. We use two types of transfer learning: pretraining and domain adaptation. Accordingly, we divide the training process of RISE-DT into two phases: pretraining phase and fine-tuning phase (Figure 1). In both of the phases, the model takes the elevator data as the input and performs transfer learning from source scenarios to target scenarios. Let S be the source elevator scenario and T be the target elevator scenario. The goal is to improve T with knowledge transferred from S. In the pretraining phase, we first pretrain the original model of RISE-DT with source elevator scenarios $S^1_{pre}, S^2_{pre}, ..., S^N_{pre}$ and target elevator scenarios $T^1_{pre}, T^2_{pre}, ..., T^N_{pre}$, instead of S and T. In the fine-tuning phase, we perform transfer learning from S to T with the pretrained RISE-DT.

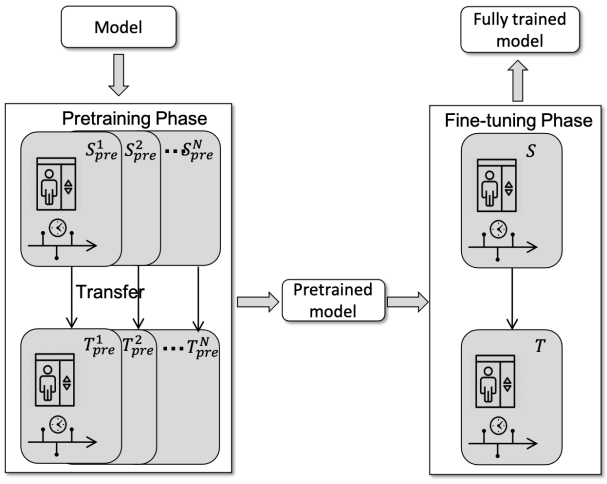

Figure 1: Overview of RISE-DT

As shown in Figure 2, the design of RISE-DT mainly involves three main techniques: *DT*, *transfer learning* and *UQ*, and the overall process builds two separate DTs for the source and target elevator scenarios. For the source DT, we first perform **UQ** to select *Most Uncertain Samples* and feed them into the source DT. We then use **transfer learning** to train the target DT. In the following subsections, we provide details about DT in Section 3.1, followed by the use of transfer learning and UQ, in Sections 3.2 and 3.3.

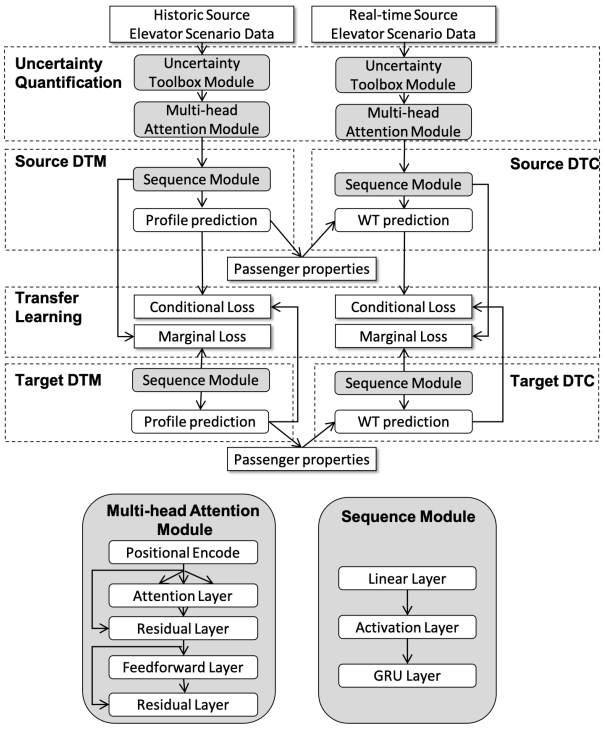

Figure 2: Architecture of RISE-DT. *DTM*, *DTC* and *WT* denote DT Model, DT Capability, and Waiting Time, respectively.

3.1 Digital Twin

As shown in Figure 2, RISE-DT's DT has two key components: the *DTM* and the *DTC*. The goal of the DTM is to simulate the real world elevator scenario by predicting/generating passenger properties (e.g., destination, weight), while the DTC tries to predict waiting time with the input of both real-world passenger properties and generated passenger properties from the DTM. The generated passenger proprieties can provide the DTC with more information because these two types of inputs are different despite of the same format due to two main reasons discussed below.

First, the DTM is trained with historical data, while the DTC is trained with only real-time data. After training, the parameters of the DTM converge at local optimal. This process incorporates information of the historical data into these trained parameters, and in turn passes to the generated passenger properties. As a result, using generated passenger properties as input introduces more information (from the historical data) into the DTC.

Second, the DTM simulates the elevator's behavior. Ideally, it should accurately predict how the elevator should operate at any given time if it is in a normal state. This process inevitably introduces a learning bias which induces the DTM to assume the elevator system is operating normally. However, the real passenger properties may lead to abnormality, causing a discrepancy between

real and generated passenger properties. Therefore, we argue that generated passenger properties focus on different aspects compared to real passenger properties. The DTC takes both generated and real passenger properties as input and makes predictions based on their own features as well as the discrepancy between them.

Given that elevator scenario data is in a time series form, the literature has shown the superiority of sequence models for handling such data [15]. Examples include Recurrent Neural Networks (RNNs) [38] and Gated Recurrent Unit networks (GRUs) [11]. We performed a preliminary analysis comparing RNN and GRU with a subset of SWaT as the validation dataset, and observed that GRU outperforms RNN in predictive performance. Hence, we employ GRUs in both the DTM and DTC to capture temporal characteristics of a scenario. The GRU-based DTM simulates the real-world operation of elevator scenarios by predicting/generating passenger properties, containing information about a passenger's loading/unloading time, weight, capacity, arrival floor, destination floor, etc. Such a generated profile, along with real-time elevator scenario data, are fed into the GRU-based DTC component of RISE-DT, which hence predicts waiting time for each passenger.

3.1.1 Digital Twin Model. DTMs are often built as behavior models [53]. However, considering that sequence models (e.g., GRU) have shown their successes in various domains, we design our DTM as a GRU-based neural network. This model consists of four layers: the linear layer, activation layer, GRU layer, and prediction layer. Let u_t be the passenger's property vector at time point t.

Linear layer transforms the passenger's property (u_t) vector into hidden vectors (x_t). In Equation 1, W and b are the weight matrix and bias vector.

$$x_t = W_{dtm} \cdot u_t + b_{dtm} \tag{1}$$

Activation layer applies the ReLU activation function to add non-linearity on the hidden vectors as in Equation 2.

$$x_t = relu(x_t) \tag{2}$$

GRU layer introduces two types of gates into the DTM: update gate z and reset gate r. Let $h_t \in R^H$ be the hidden state vector of GRU and W, U, b are all weight matrices. The update rules of GRU are as follows:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{3}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{4}$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_h) \tag{5}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \hat{h}_t \tag{6}$$

Prediction layer transforms the hidden vectors into the label space. A single passenger has multiple properties. We build a separate prediction layer for each property (Figure 2). But they are trained at the same time in a multi-task learning manner.

Loading time prediction. Loading time denotes time needed for a passenger to get in an elevator. Predicting loading time as a regression task potentially leads to over-fitting due to limited data volume. We simplify this prediction by discretizing loading time into 10 categories and convert this problem into a classification problem. Equation 7 shows the linear prediction layer for loading time. $W_{loading} \in R^{H \times 10}$ is the weight matrix.

$$loading_t = W_{loading} h_t + b_{loading} \tag{7}$$

Unloading time prediction. Unloading time denotes time needed for a passenger to get off an elevator. Similar to loading time, we also discretize unloading time into 10 categories. Equation 8 shows the linear prediction layer for unloading time. $W_{unloading} \in R^{H \times 10}$ is the weight matrix.

$$unloading_t = W_{unloading} h_t + b_{unloading} \tag{8}$$

Weight prediction. Similar to loading time prediction, we convert this continuous prediction task into a less fine-grained classification task with only 5 category labels to avoid over-fitting. Equation 9 shows the linear prediction layer for passenger weight. $W_{weight} \in R^{H \times 5}$ is the weight matrix.

$$weight_t = W_{weight} h_t + b_{weight} \tag{9}$$

Capacity prediction. The capacity denotes passengers' willingness to get in an elevator based on the percentage of the elevator occupied with passengers. For example, a passenger with a capacity value of 0.6 will not enter, when an elevator 60% or more full. To prevent over-fitting, we convert it into a 5-category classification task. Equation 10 shows the linear prediction layer for the passenger capacity. $W_{capacity} \in R^{H \times 5}$ is the weight matrix.

$$capacity_t = W_{capacity} h_t + b_{capacity} \tag{10}$$

Arrival floor prediction. The arrival floor denotes which floor a specific passenger arrives and makes a call. This property is discrete by nature. This prediction layer is designed as a N categories classification problem, where N denotes the number of floors this building has. Equation 11 shows the linear prediction layer for the arrival floor. $W_{arrival} \in R^{H \times N}$ is the weight matrix.

$$arrival_t = W_{arrival} h_t + b_{arrival} \tag{11}$$

Destination floor prediction. The destination floor denotes the floor a specific passenger intends to go. This property is also discrete and can been designed as a N categories classification problem similar as the arrival floor prediction layer. Equation 12 shows the linear prediction layer for the destination floor. $W_{destination} \in R^{H \times N}$ is the weight matrix.

$$destination_t = W_{destination} h_t + b_{destination} \tag{12}$$

Combining all these properties, we acquire complete passenger properties $\hat{x_t}$ with the help of the DTM. Then we feed $\hat{x_t}$ along with real-time data into the DTC for the prediction of waiting time.

3.1.2 Digital Twin Capability. The DTC of RISE-DT predicts the waiting time of a passenger by employing GRU. As for the DTM, the DT capability has four layers: the linear, activation, GRU, and waiting time prediction layers.

Linear layer concatenates the generated passenger profile vector $\hat{x_t}$ and real-time profile x_t as an input vector $u_t \in R^H$, then transforms it into the hidden space:

$$u_t = W_{dtc} \cdot u_t + b_{dtc} \tag{13}$$

Activation layer increases the DTC's non-linearity with ReLU:

$$u_t = relu(u_t) \tag{14}$$

GRU layer captures temporal characteristics of elevator data, and introduces updating rules in Equation 3.
Prediction layer makes predictions about waiting time for each passenger, which is a continuous task.

3.2 Uncertainty Quantification

DT (Section 3.1) can potentially perform well with sufficient labeled data. However, we rarely have access to abundant labeled data from the target elevator scenario. Therefore, RISE-DT has to learn from a source elevator scenario, where data are not all valuable in terms of improving the prediction performance of RISE-DT in target scenario. This inspires us to utilize *UQ* for selecting the most uncertain samples for transfer learning. The overview of the UQ component of RISE-DT is presented in Figure 2, which contains two types of modules: *Uncertainty Toolbox Modules* and *Multi-head Attention Modules*.
Uncertainty Toolbox is a tool for predictive UQ. In RISE-DT, we use it to calculate two metrics for each sample: calibration c and sharpness s. We normalize these two metrics into the range of (0,1), making the addition of these metrics feasible. We define a new metric m as the weighted sum of these two metrics, as shown in Equation 15, where λ is a hyperparameter.

$$m = \lambda normalize(c) + (1-\lambda)normalize(s) \tag{15}$$

With this metric, we can rank all the samples we have and select M samples with the highest uncertainty scores with Equation 16

$$U' = top([u_0, u_1, ..., u_n]) \tag{16}$$

Multi-head Attention. Though with UQ, we identify high uncertain (therefore most valuable) samples in the source domain, we cannot completely ignore other samples. Doing so will lose information about context and time. Therefore, we use a multi-head attention module [57] to encode this information into sample vectors. This attention module has two sub-modules: attention module and feed-forward module.
Attention Module takes most uncertain samples U' as the input and uses positional encoding to incorporate position information as shown in Equation 17.

$$U' = U' + position\_encoding(U') \tag{17}$$

We duplicate this vector into three identical vectors and perform a linear transformation on them. The outputs of this transformation are: U_k, U_v and U_q. U_k and U_v are for calculating the attention weight of each vector. With this weight, we then calculate a weighted sum of U_q. Equation 18 shows the calculation of attention.

$$attention\_weight = U_v U_v^T \tag{18}$$

$$U_{attention} = attention\_weight U_v \tag{19}$$

Then, a residual connection between the input vectors and the output of attention module is built, as shown in Equation 20.

$$U_{attention} = normalize(U' + U_{attention}) \tag{20}$$

Feed-forward Module takes the output of attention module as its input and feeds it into a feed-forward network (Equation 21).

$$U = W_{attention} U_{attention} + b_{attention} \tag{21}$$

$$U_{feed} = relu(U) \tag{22}$$

Then, a residual connection between the attention output and the feed-forward output is built (Equation 23).

$$U_{out} = normalize(U_{attention} + U_{feed}) \tag{23}$$

3.3 Transfer Learning

With the encoded samples generated in Section 3.2, transfer learning transfers knowledge from the source elevator scenario to the target one, by aligning vectors. As shown in Figure 2, transfer learning calculates two types of losses: marginal and conditional distribution loss, which align data distributions and prediction distributions, respectively. Reducing these two losses brings both the source and target vectors into an intermediate space, which represents the common knowledge shared between the two elevator scenarios.

For marginal distribution loss, we calculate the KL-divergence of the output vectors from the GRU layers of the source and target domains, as shown in Equation 24.

$$loss_{KL} = \sum u_i^{source} log u_i^{target} \tag{24}$$

For conditional distribution loss, we calculate the Maximum Mean Discrepancy (MMD) of the prediction layer outputs from the source and target domains, as shown in Equation 25.

$$loss_{MMD} = || \frac{1}{n^s} \sum_{i=1}^{n^s} u_i^s - \frac{1}{n^t} \sum_{i=1}^{n^t} u_i^t || \tag{25}$$

$$= Tr(UMU^T) \tag{26}$$

These two losses are then summed as the final loss (Equation 27).

$$loss = loss_{KL} + loss_{MMD} \tag{27}$$

4 EXPERIMENT DESIGN

To evaluate RISE-DT, we propose four research questions (RQs), as discussed in Section 4.1, followed by the introduction of the subject system employed (Section 4.2). Section 4.3 presents the evaluation metrics and statistical tests used in this paper. Section 4.4 discusses how we chose hyperparameters and about experiment execution.

4.1 Research Questions

The four RQs are described in Table 1. RQ1 evaluates if RISE-DT is effective in terms of transferring knowledge from a source traffic template to a target traffic template. With RQ2, we aim to see if RISE-DT is effective in handling cases in which we only have access to data generated with an earlier version of an industrial elevator dispatcher but need to learn a prediction model for the latest version of the dispatcher. The application contexts of both RQ1 and RQ2 are commonly seen in the real-world operation of elevators. RQ3 evaluates the effectiveness of introducing UQ to RISE-DT, i.e., using UQ to select uncertain samples from the source data and input them to transfer learning. RQ4 considers the practical usefulness of RISE-DT by evaluating its time cost.

4.2 Subject System

The subject system is from Orona, who provided us with 10 versions of an elevator dispatcher. Each version is considered as an individual *dispatcher* in the rest of the paper. We deployed them in Elevate, which simulates the procedure of passengers arriving at

Table 1: Experiment Design

RQ	Metric	Scenarios	
		Type	**Instance**
RQ1: How effective is RISE-DT when transferring knowledge across scenarios with different traffic templates?	MSE	UpBest → LunchBest; LunchBest → UpBest	1
RQ2: How effective is RISE-DT when transferring knowledge across scenarios with different versions of an elevator dispatcher?	MSE	UpWorse → UpBest; LunchWorse → LunchBest	10
RQ3: Is UQ effective in transfer learning?	MSE	UpBest → LunchBest; LunchBest → UpBest; UpWorse → UpBest; LunchWorse → LunchBest	11
RQ4: Is RISE-DT practically useful in terms of time cost?	Time	UpBest → LunchBest; LunchBest → UpBest; UpWorse → UpBest; LunchWorse → LunchBest	11

floors, making calls, boarding arrived elevators, making car calls, and arriving destinations. Elevate provides interfaces that allow us to simulate various elevator scenarios. We consider the following four elevator scenarios: **LunchBest** (or **LunchWorse**) denoting the scenario of the best (or a worse) elevator dispatcher operating during lunch peak (12:15pm - 13:15pm), while **UpBest** (or **UpWorse**) denoting the best (or a worse) elevator dispatcher operating during up peak (8:30am - 9:30am).

4.3 Metrics and Statistical Tests

4.3.1 Mean Squared Error (MSE). Since RISE-DT predicts the waiting time for each passenger, we employ MSE– a commonly used metric for regression prediction tasks, which computes the average square of errors. Let Y be the vector of observed values and $\hat{Y}$ be the predicted value. MSE is defined as in Equation 15.

$$MSE = \frac{1}{n}\sum_{i-1}^{n}(Y_i - \hat{Y}_i)^2 \tag{28}$$

4.3.2 Uncertainty. To answer RQ2, we need to calculate the uncertainty of each neural network model of the DT corresponding to each dispatcher with Equation 29.

$$uncertainty(dispatcher\ i) = \frac{\sum_{j=0}^{n-1} m_j}{n} \tag{29}$$

where m_j denotes the uncertainty value for sample j as in Equation 15. The uncertainty of each sample is calculated based on the prediction distribution produced by the neural network model of the DT of each dispatcher. Hence, the calculated sample uncertainty can also reflect the uncertainty of the dispatcher.

4.3.3 Training Time. We used training time for the evaluation of cost. To this end, we consider two types of training time: pretraining time and fine-tuning time, as RISE-DT has these two phases (Figure 1). Let S be the source elevator scenario and T be the target elevator scenario for transfer learning. Pretraining of RISE-DT is performed on N scenarios that do not include S and N. We define convergence time for one transfer as in Equation 30.

$$time_{convergence} = time_{early\_stopping\_end} - time_{start} \tag{30}$$

In Equation 31, we calculate the pretraining time by summing the convergence time on each single transfer.

$$time_{pretrain} = \sum_{i=0}^{N} time_{convergence}(S_i, T_i) \tag{31}$$

where $time_{early\_stopping\_end}$ denotes the early stopping time point (no improvement for 5 consecutive epochs) and $time_{start}$ denotes the starting time point of training. Equation 32 defines the fine-tuning time as convergence time on source S and target T.

$$time_{finetune} = time_{convergence}(S, T) \tag{32}$$

4.3.4 Statistical Testing. To deal with randomness in neural network training, we repeat each experiment 30 times and perform the Mann-Whitney U test [2] to study the statistical significance of the improvements. We test all the pair-wise comparisons (*Method A* and *Method B*) in each RQ. The null hypothesis is that there is no significant difference between the two methods. If the null hypothesis is rejected, we conclude that *Method A* and *Method B* are not equivalent. Furthermore, we follow the suggestions in [2] and choose the Vargha and Delaney's A12 as the effect size. A12 shows the chances that *Method A* will get better results than *Method B*. If A12 is greater than 0.5, we can conclude that *Method A* has a higher chance of getting better results than *Method B*, and vice versa. We also use the Spearman's rank correlation coefficient to evaluate the correlation between dispatcher uncertainty and MSE [27] (Section 5.5). Spearman correlation is a non-parametric measure which is calculated with the rankings of two variables. The correlation coefficient tends to be high if observations from the two variables have a similar rank relative position label and vice versa. This value takes values between −1 and 1.

4.4 Settings and Execution

Manually assigning hyperparameter values can introduce bias. To reduce such bias, we performed a 10-fold cross validation to select a best hyperparameters: splitting the dataset into 10 chunks sequentially, with the first 9 of them used for training and the last for validation. As a result, we set the weight of calibration and sharpness λ as 0.9, the hidden size as 350, and used the bidirectional 2 layers GRU in both of the DTM and DTC of RISE-DT. The neural network layers were built with the PyTorch framework [41]. The experiments were performed on one node from Simula Research Laboratory's eX3 infrastructure with 2x Intel Xeon Platinum 8186, 1x NVIDIA V100 GPUs.

5 RESULTS AND ANALYSES

In this section, we provide results and analyses for answering each RQ, followed by the threats to validity.

5.1 RQ1–Transferring Knowledge across Scenarios With Different Traffic Templates

We compare RISE-DT with and without transfer learning in two setups: from scenario UpBest to LunchBest and from LunchBest to UpBest. We observe from Table 2 that transfer learning from UpBest to LunchBest improves MSE by 26.97 (149.08-122.13), whereas from LunchBest to UpBest the MSE improvement is 21.350 (153.49-132.14). The p-values for both scenarios are smaller than $1e-3$ suggesting that RISE-DT with transfer learning is significantly better than RISE-DT without transfer learning in the context of transferring across traffic templates, which we call *traffic-template-variant transfer* in short. The A12 values show that RISE-DT with transfer learning has a higher probability of yielding better results than RISE-DT without transfer learning since both A12 values are less than 0.5. Furthermore, with UQ, MSE is further reduced (see Table 2) for both scenarios. The top two boxplots in Figure 3 show the MSE distributions of the two traffic-template-variant transfers. We observe that the distributions with transfer learning have less variance than those without transfer learning, suggesting that with UQ the transfer of knowledge seems more reliable.

Conclusion: RQ1

We observe an improvement in terms of MSE and a reduction of variance when comparing RISE-DT without and with traffic-template-variant transfer learning. Thus, RISE-DT with transfer learning can effectively transfer knowledge across elevator DTs built for different traffic templates.

Table 2: Results for traffic-template-variant transfer learning for two scenarios (RQ1). *W/o* and *W* denote without and with.

	UpBest->LunchBest			LunchBest->UpBest		
	Mean	**p-value**	**A12**	**Mean**	**p-value**	**A12**
W/o transfer	149.08	-	-	153.49	-	-
W/ transfer	122.10	0	0	132.14	0	0.14
W/ UQ	117.95	0	0.14	130.33	1.7e-3	0.28

5.2 RQ2–Transferring Knowledge across Scenarios With Different Dispatchers

Table 3 shows the results of transferring knowledge across scenarios with different dispatchers, which we call dispatcher-variant transfer learning in short. We performed experiments on 10 elevator scenarios with worse performance than our best dispatcher. The third row in Table 3 presents the MSE metric of RISE-DT on the Uppeak (149.076) and Lunchpeak (153.49) traffic templates of the best dispatcher. We performed 10 separate experiments, transferring knowledge from 10 worse performing dispatchers to the best one. By doing so, the best dispatcher can gain knowledge about worse-performing dispatchers' specific cases that are rarely seen in their own scenario data. We can observe from Table 3 that transfer learning from all the 10 dispatchers to the best one brings a reduction of MSE with the Uppeak traffic template by 15.58 (149.08-133.50) and Lunchpeak traffic template by 15.57 (153.49-137.92), on average. With Uppeak, transferring from dispatcher 2 reaches the lowest MSE as 122.07, while transferring from dispatcher 7 reaches the lowest MSE as 130.904 with Lunchpeak. Statistical testing results for these dispatcher-variant transfers show that most p-values are smaller than 1e-2 (except for dispatcher 5 with Lunchpeak), indicating that transfer learning from the worst-performing dispatchers to the best one improves MSE significantly. Most of the A12 results are also strong, i.e., much lower than 0.5. One plausible explanation is because of the increased data volume. More specifically, with transfer learning, RISE-DT takes advantage of data from elevator scenarios that are previously unavailable, which, as similar to data augmentation, increases data volume and hence potentially improves the performance of deep neural networks [49]. Moreover, in our context, best-performing dispatchers refer to newer versions of dispatchers, which does not necessarily mean that a best-performing dispatcher has all the information of a worst-performing dispatcher.

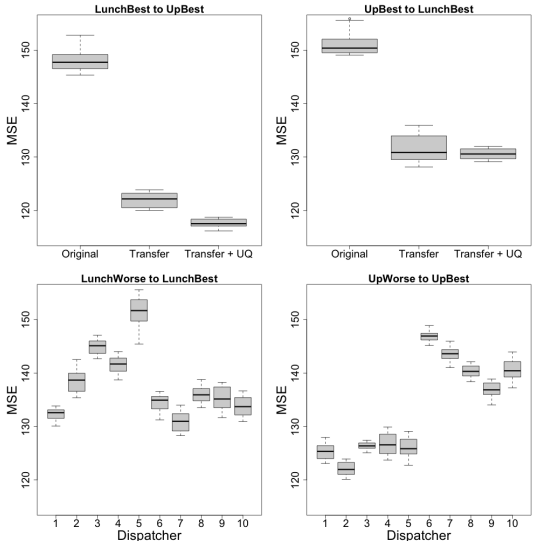

Figure 3: MSE results (in boxplots) of the 30 runs of the traffic-template transfer learning (RQ1) and the dispatcher-variant transfer learning (the bottom two figures) (RQ2).

Conclusion: RQ2

We observe an improvement of MSE for both the Uppeak and Lunchpeak traffic templates, when comparing RISE-DT with and without dispatcher-variant transfer learning. Thus, RISE-DT can effectively transfer knowledge across different elevator DTs built corresponding to different dispatchers.

5.3 RQ3–Effectiveness of UQ

We evaluate the effectiveness of UQ in both traffic-template-variant and dispatcher-variant transfer learning. From the last row of Table 2, we can observe that, for traffic-template-variant transfer learning, UQ improves transfer learning from UpBest to LunchBest by

4.16 (122.10-117.95) and from LunchBest to UpBest by 1.81 (132.14-130.33), respectively. Statistical test results show that both improvements are statistically significant, with confidence level equal to 99%, p-values smaller than $1e-3$, and strong A12 values (being 0.14 and 0.28, respectively, which are lower than 0.5).

Table 3 shows that UQ generally improves dispatcher-variant transfer learning on average by 4.46 (133.50-129.04) and 4.24 (137.92-133.68), for the Uppeak and Lunchpeak, respectively. Transferring from dispatcher 2 to our best dispatcher reaches the lowest MSE as 120.41 for Uppeak, while for Lunchpeak, dispatcher 10 achieved the lowest MSE. The results are statistically significant for most of the cases except for dispatcher 5 with Lunchpeak. This means that UQ is effective in boosting transfer learning's performance in dispatcher-variant transfer learning. Moreover, the A12 values mostly are much lower than 0.5 (except for dispatcher 5 with Lunchpeak), telling that RISE-DT with UQ has a better chance of yielding better MSE.

Conclusion: RQ3

We observe significant improvement of MSE, hence, the benefit of introducing UQ to transfer learning across traffic templates and dispatchers.

Table 3: Results for dispatcher-variant transfer learning (RQ2 and RQ3). The lowest values are highlighted in bold. *Disp* denotes dispatcher; *W/o* and *W* denote without and with UQ.

		Uppeak			Lunchpeak		
Disp	**UQ**	**MSE**	**p-value**	**A12**	**MSE**	**p-value**	**A12**
Best	W/o	149.08	-	-	153.49	-	-
1	W/o	125.38	<1e-3	0	132.26	<1e-3	0
	W	122.22	<1e-3	0.03	131.10	3e-3	0.28
2	W/o	**122.07**	<1e-3	0	138.75	<1e-3	0
	W	**120.41**	<1e-3	0.14	131.40	<1e-3	0
3	W/o	126.37	<1e-3	0	144.96	<1e-3	0
	W	124.95	<1e-3	0.16	141.14	<1e-3	0.05
4	W/o	126.83	<1e-3	0	141.56	<1e-3	0
	W	120.85	<1e-3	0	137.44	<1e-3	0.01
5	W/o	126.12	<1e-3	0	151.27	0.21	**0.43**
	W	121.87	<1e-3	0	147.79	<1e-3	0.29
6	W/o	146.90	<1e-3	0.30	134.40	<1e-3	0
	W	135.66	<1e-3	0	131.01	<1e-3	0.01
7	W/o	143.54	<1e-3	0	**130.90**	<1e-3	0
	W	139.68	<1e-3	0.01	129.27	<1e-3	0.27
8	W/o	140.27	<1e-3	0	136.02	<1e-3	0
	W	132.39	<1e-3	0	132.92	<1e-3	0.04
9	W/o	136.96	<1e-3	0	135.34	<1e-3	0
	W	135.43	0.001	0.26	128.09	<1e-3	0
10	W/o	140.53	<1e-3	0	133.71	<1e-3	0
	W	136.98	<1e-3	0	**126.69**	<1e-3	0
Average	W/o	133.50	-	-	137.92	-	-
	W	129.04	-	-	133.69	-	-

5.4 RQ4–Time Cost

We measure the time cost of the pretraining and fine-tuning phases of RISE-DT. Table 2 shows results for the four transfers: UpBest to LunchBest and LunchBest to UpBest for traffic-template-variant transfer learning, and UpWorse to UpBest and LunchWorse to LunchBest for dispatcher-variant transfer learning.

The pretraining times for all the four transfers are between 50.1 and 65.4 hours, which are practically acceptable considering that pretraining is performed only once. In the fine-tuning phase, column *w/o Transfer* shows the time required to build the DT directly from elevator scenario data. Columns *Transfer* and *Transfer+UQ* show the time spent on transferring knowledge of the DT with and without UQ. RISE-DT without transfer learning took the maximum of 4.1 hours (LunchBest->UpBest), while RISE-DT with transfer learning (but without UQ) took only 1.4 hours in the worst case (LunchWorse->LunchBest). We argue that this reduction of time cost is due to the transferred knowledge from other elevator scenarios, which helps RISE-DT find the optimal parameters faster compared to random initialization. The time cost for transfer learning with UQ is increased as applying it introduces a complex neural network module (i.e. the Multi-head attention module, Figure 2).

Conclusion: RQ4

Time cost of RISE-DT is practically acceptable and lower than time required for building DTs from scratch, i.e., without using transfer learning. Thus, deploying RISE-DT in real world is feasible in terms of time cost.

Table 4: Results of time cost in hours (RQ4)

	Pretraining	Fine-tuning		
Transfer	**Time**	**w/o Transfer**	**w/ Transfer**	**Transfer + UQ**
UpBest -> LunchBest	62.3h	3.7h	1.2h	1.5h
LunchBest -> UpBest	65.4h	4.1h	1.2h	1.5h
UpWorse -> UpBest	58.5h	2.9h	1.1h	1.3h
LunchWorse -> LunchBest	50.1h	2.7h	1.4h	1.9h

5.5 Discussion

Transfer Learning–An Effective Mechanism to Transfer Knowledge across DTs. The results of RQ1 and RQ2 (Section 5.1 and Section 5.5) showed that RISE-DT is effective on performing both traffic-template-variant and dispatcher-variant transfer learning. There are two main reasons. First, transfer learning introduces more data for training, which, to a certain extent, alleviates over-fitting in neural networks due to insufficient training data. We have access to several elevator scenarios, but each elevator scenario differs in either dispatcher or traffic template. As a result, this difference leads to discrepancies in elevator scenario data distribution. Directly training target RISE-DT with data from other elevator scenarios can cause a reduction of performance due to this discrepancy. Transfer learning elegantly tackles this challenge by aligning the source and target data distribution in an intermediate space (Section 3.3). In particular, we align the data by reducing both marginal and conditional loss, which, as a result, generates a new set of vectors with minimal discrepancy between the source and target data. These vectors identify the common knowledge shared between source and target data, and thus complete the transfer learning. Second, the source elevator scenario is different from the target elevator scenario, but related; therefore, transfer learning can find common knowledge shared between them. Such common knowledge increases the generalization ability in our model, as

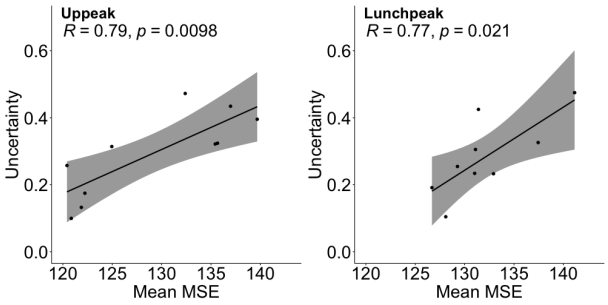

Figure 4: Correlations of mean MSE and uncertainty

learning from multiple sources of data prevents models from converging quickly on a single dataset [10].

Uncertainty Quantification (UQ)–An Effective Mechanism to Improve the Accuracy of Knowledge Transfer across DTs. The results of RQ3 (Section 5.3) show that UQ further improves the effectiveness of transfer learning. Though UQ was mainly developed to evaluate a model's robustness [1], we use it to select the most uncertain samples since such samples contain *important* (in terms of uncertainty) information that can improve the effectiveness of transfer learning. This idea is very similar to *importance sampling* in statistics, which has been applied in deep learning for optimizing RNNs, CNNs, etc. [30], where weights are assigned based on the variance of each sample. In the future, in addition to UQ, we will investigate other importance sampling mechanisms to further enhance the effectiveness of transfer learning.

Strong Correlation between MSE and Uncertainty. We noted that the MSE values vary from dispatcher to dispatcher. We make an assumption that this difference is correlated with dispatchers' uncertainties. To test the assumption, we calculated the uncertainty for each dispatcher with Equation 29, and plotted the correlation in Figure 3. The Spearman's correlation rank values for Uppeak and Lunchpeak are 0.79 and 0.77, respectively, indicating a strong positive correlation between uncertainty and MSE. The p-values are 0.00096 and 0.014, which indicate that the correlations are statistically significant. The grey areas in both figures present the 95% confidence interval. In both figures, only two data points fall out of the grey area. Based on this result, we can then recommend to calculate the uncertainty of a dispatcher with experimental data before its deployment, which could potentially indicate its MSE in operation and thus improve the safety and security of the elevator system at an early stage.

Practical Implications. First, our work is valuable for Orona since the DTs developed with RISE-DT can be used as a prior knowledge to test or assess a new dispatcher before deploying it in the real operation. Thus, DTs built with transferred knowledge will facilitate the evolution and maintenance of new dispatcher versions. Second, transferring knowledge of DTs for different elevator scenarios will reduce the development and maintenance cost of DTs, as our experiments also demonstrated. Third, our results with UQ provide evidence showing that uncertainty can affect the predictions made with DTs; thus, the industry shall consider uncertainties explicitly during the design, development, and operation of elevator systems and their corresponding DTs.

5.6 Threats to Validity

We identify four key threats to the validity. First, our experiments were conducted on a simulator of elevator. Though the interfaces provided by this simulator allow us to choose from real industrial elevator dispatcher and elevator scenarios, it is SiL, thus different from elevator systems operating in the real world. However, using Elevate is the current practice of our industrial partner Orona and SiL is a common practice in many domains. Second, we use transfer learning and UQ. For transfer learning, we align data from the source and target elevator scenario. For UQ, we use Uncertainty Toolbox. We are aware that there are other methods and techniques that could accomplish transfer learning (e.g. Bayesian methods [35]) and UQ (e.g. Uncertainty Wizard [61]). In the future, we will experiment with such techniques. Third, we use only MSE for evaluating the effectiveness of the waiting time prediction. We choose this metric because this is a commonly used metric in regression prediction tasks. However, we are aware of other metrics such as Root Mean Square Error (RMSE) [9] and Coefficient of determination (R^2 score) [29], which will be investigated in the future. Fourth, the core neural network used in our DT building is GRU. We know that there are other sequences models, e.g., RNN [48] and LSTM [48]. We performed preliminary experiments on these sequence models but results show that GRU performs better.

6 RELATED WORK

We discuss the related work from three aspects: CPS and DTs (Section 6.1), transfer learning (Section 6.2) and UQ (Section 6.3).

6.1 Cyber-physical Systems and DTs

Passenger waiting time has already been an important QoS indicator of elevators, there have been several works proposed for reducing it by optimizing elevator scheduling algorithms [17, 22, 54] with dynamic fuzzy logic, evolution algorithm, etc. To compare with these works, our goal is about predicting passenger waiting time, not reducing/optimizing it, which, we consider, is about evolving industrial elevators themselves.

When looking at CPS, in general, they are typically susceptible to risks from the physical and cyber spaces. To mitigate such risks, many security and safety enhancement techniques have been proposed, e.g., [5, 28, 37, 43]. Along with the increased deep learning use to enhance CPS security and safety [31, 50, 62], more and more researchers and practitioners realize that obtaining labeled data for real world CPS is very expensive, even infeasible in some contexts. The scarcity of labeled data hinders the training process of deep learning models. Thus, in this paper, we follow this research line by designing RISE-DT as a deep learning method. RISE-DT introduces transfer learning and UQ to mitigate the challenge of the scarcity of labeled data in the CPS domain.

Moreover, security and safety risks evolve over time. The literature mostly focus on statically training models with offline CPS log data (e.g., [26, 46]), which is vulnerable to previously-unknown attacks or faults [36]. CPS in operation, continuously generates data, which can potentially evolve a statically generated method to identify emerging security and safety issues. However, most existing methods can not take advantage of this newly generated data without full-scale retraining. DT technologies bring a novel

way to overcome this challenge by synchronizing with CPS in real-time [6, 7, 14, 18–20, 55]. Particularly, Bécue et al. [6] proposed to use DTs for analyzing how CPS should be engineered under attack. Eckhart et al. [18] equipped DTs with logic and network features, for analyzing if an attacker can compromise programmable logic controllers. Bitton et al. [7] proposed to perform tests on a DT, instead of real CPS. Damjanovic-Behrendt [14] used DTs for privacy assessment of real smart car systems. These works show the superiority of DT technologies. But, to the best of our knowledge, we are the first to focus on building DTs for elevator systems waiting time prediction.

6.2 Transfer Learning

There are four strategies for transfer learning. The *model control strategy* performs transfer learning at the model level. For instance, Duan et al. [16] proposed Domain Adaptation Machine (DAM), which uses multiple source domain data, builds a classifier for each domain, and adopts regularizers to control the complexity of the final model. The *parameter control strategy* assumes that the parameters of a model reflect the knowledge it has learned. For instance, Zhuang et al. [69] proposed a transfer learning approach for text classification, which shares parameters directly between the source and target models. The *model ensemble strategy* performs transfer learning by combining several source models together. For example, Gao et al. [23] proposed to train several weak classifiers of different model structures on multiple source domains and compute the final model as a weighted vote of these weak classifiers. *Deep learning transfer techniques* transfer knowledge between two deep learning models, by aligning the representations of corresponding layers from the source and target models. Along this line, Zhuang et al. [69] proposed transfer learning with autoencoder, which aligns reconstruction, distribution and regression representations. Tzeng et al. [56] extended this method by adding an adaptation layer. Long et al. [34] performed this alignment in multiple layers in their model Deep Adaptation Networks.

In conclusion, model control and parameter control strategies are early strategies that perform knowledge transfer with intuitive methods such as adding regularizer and sharing parameters. Their performance is as good as model ensemble and deep learning transfer techniques. Model ensemble works the best with multiple heterogeneous source domains and requires a lot of computing resources. Deep learning transfer techniques works for transferring knowledge between two neural network models. Since our RISE-DT is a neural network-based DT, we follow this research line, and align the representation of the GRU layer and prediction layer.

6.3 Uncertainty Quantification and Analyses

Many UQ methods are based on Bayesian techniques. For instance, Wang et al. [59] proposed to use the probability theory to interpret parameters of neural networks. Later on, Srivastava et al. [51] used Monte Carlo dropout as a regularization term for the prediction uncertainty computation to avoid posterior probability calculation. Salakhutdinov et al. [44] proposed a stochastic gradient Markov chain Monte Carlo (SG-MCMC) method, which only needs to estimate the gradient on small sets of mini-batches requiring far less computing than estimating the posterior distribution directly. Neural networks are also being used for estimating the posterior distribution. For instance, Ghosh et al. [24] proposed a variational autoencoder (VAE) with an encoder and decoder both having the neural network structure. Other UQ methods include deep Gaussian processes [13] and ensemble-based UQ [33].

Several open-source UQ tools are available to use. Uncertainty wizard [61] is a TensorFlow Keras plugin supporting common quantification methods, e.g., Bayesian and ensemble-based methods. Uncertain toolbox [12] is built on Pytorch also providing commonly applied Bayesian and ensemble UQ methods, along with other metrics such as calibration, sharpness and accuracy. We opted for Uncertainty Toolbox because our model is coded with the Pytorch library, which makes it easier to use a Pytorch-based tool like uncertainty toolbox.

Generic UQ methods encourage researchers to employ them in specific application domains. For instance, Catak et al. [8] proposed NIRVANA for prediction validation of deep learning models based on uncertainty metrics. In addition, in the domain of CPS, some methods have been proposed to enable uncertainty-aware analyses. For instance, Han et al. [32] proposed an approach to systematically classify uncertainties with the Cynefin framework and assess the robustness of industrial elevator systems based on uncertainty classification results. Zhang et al. [65–68] proposed a series of methods in dealing with CPS uncertainties.

7 CONCLUSION AND FUTURE WORK

We propose RISE-DT to build DTs of industrial elevators with neural networks to support the evolution of DTs. RISE-DT employs transfer learning with Uncertainty Quantification (UQ) to learn knowledge from source elevator scenario and perform accurate predictions on target elevator scenario with DT. We conducted experiments with four different types of scenarios and 10 different dispatchers. The results showed an average reduction of mean square error of 13.131% with transfer learning and a further reduction of 2.71% with UQ, which proves the effectiveness of both transfer learning and UQ. In the future, we will employ adversarial samples for transfer learning, which can potentially further improve the performance since more data will be included in the source domain. Moreover, we want to explore additional transfer learning techniques, DT construction methods, and UQ techniques followed by performing more comprehensive experiments.

ACKNOWLEDGMENTS

Qinghua Xu is supported by the security project funded by the Norwegian Ministry of Education and Research. Shaukat Ali and Tao Yue are supported by Horizon 2020 project ADEPTNESS (871319) funded by the European Commission and project Co-tester (#314544) funded by Research Council of Norway.

REFERENCES

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* 76 (2021), 243–297. https://doi.org/10.1016/j.inffus.2021.05.008 arXiv:2011.06225

[2] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. *Proceedings -*

International Conference on Software Engineering (2011), 1–10. https://doi.org/10.1145/1985793.1985795

[3] Aitor Arrieta, Jon Ayerdi, Miren Illarramendi, Aitor Agirre, Goiuria Sagardui, and Maite Arratibel. 2021. Using Machine Learning to Build Test Oracles: an Industrial Case Study on Elevators Dispatching Algorithms. 30–39. https://doi.org/10.1109/AST52587.2021.00012

[4] Jon Ayerdi, Sergio Segura, Aitor Arrieta, Goiuria Arratibel, and Maite Arratibel. 2020. QoS-aware Metamorphic Testing: An Elevation Case Study. 104–114. https://doi.org/10.1109/ISSRE5003.2020.00019

[5] Ayan Banerjee, Krishna Venkatasubramanian, Tridib Mukherjee, and Sandeep Gupta. 2012. Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber-Physical Systems. *Proc. IEEE* 100 (10 2012), 283–299. https://doi.org/10.1109/JPROC.2011.2165689

[6] A Bécue, Y Fourastier, I Praça, A Savarit, C Baron, B Gradussofs, E Pouille, and C Thomas. 2018. CyberFactory#1 — Securing the industry 4.0 with cyber-ranges and digital twins. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. 1–4. https://doi.org/10.1109/WFCS.2018.8402377

[7] Ron Bitton, Tomer Gluck, Orly Stan, Masaki Inokuchi, Yoshinobu Ohta, Yoshiyuki Yamada, Tomohiko Yagyu, Yuval Elovici, and Asaf Shabtai. 2018. Deriving a Cost-Effective Digital Twin of an ICS to Facilitate Security Evaluation: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I. 533–554. https://doi.org/10.1007/978-3-319-99073-6_26

[8] Ferhat Ozgur Catak, Tao Yue, and Shaukat Ali. 2022. Uncertainty-Aware Prediction Validator in Deep Learning Models for Cyber-Physical System Data. *ACM Transactions on Software Engineering and Methodology* 31 (01 2022). https://doi.org/10.1145/3527451

[9] Tianfeng Chai and R. Draxler. 2014. Root mean square error (RMSE) or mean absolute error (MAE)? *Geosci. Model Dev.* 7 (01 2014). https://doi.org/10.5194/gmdd-7-1525-2014

[10] Quanjun Chen, Xuan Song, Harutoshi Yamada, and Ryosuke Shibasaki. 2016. Learning Deep Representation from Big and Heterogeneous Data for Traffic Accident Inference. *Proceedings of the AAAI Conference on Artificial Intelligence* 30 (02 2016). https://doi.org/10.1609/aaai.v30i1.10011

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. https://doi.org/10.48550/ARXIV.1412.3555

[12] Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. 2021. Uncertainty Toolbox: an Open-Source Library for Assessing, Visualizing, and Improving Uncertainty Quantification. https://doi.org/10.48550/ARXIV.2109.10254

[13] Andreas C. Damianou and Neil D. Lawrence. 2012. Deep Gaussian Processes. https://doi.org/10.48550/ARXIV.1211.0358

[14] Violeta Damjanovic-Behrendt. 2018. A Digital Twin-based Privacy Enhancement Mechanism for the Automotive Industry. In *2018 International Conference on Intelligent Systems (IS)*. 272–279. https://doi.org/10.1109/IS.2018.8710526

[15] Ludovic Denoyer and Patrick Gallinari. 2014. Deep Sequential Neural Network. https://doi.org/10.48550/ARXIV.1410.0510

[16] Lixin Duan, Ivor Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. *Proceedings of the 26th International Conference On Machine Learning, ICML 2009* 382, 37. https://doi.org/10.1145/1553374.1553411

[17] Mahir Dursun. 2010. Estimation Of Passenger Waiting Time In Elevator Systems With Artificial Neural Network. *Intelligent Automation & Soft Computing* 16 (01 2010), 101–110. https://doi.org/10.1080/10798587.2010.10643067

[18] Matthias Eckhart and Andreas Ekelhart. 2018. Securing Cyber-Physical Systems through Digital Twins. *Ercim News* 115 (2018), 22–23.

[19] Matthias Eckhart and Andreas Ekelhart. 2018. A Specification-based State Replication Approach for Digital Twins. 36–47. https://doi.org/10.1145/3264888.3264892

[20] Matthias Eckhart and Andreas Ekelhart. 2018. Towards Security-Aware Virtual Environments for Digital Twins. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security* (Incheon, Republic of Korea) *(CPSS '18)*. Association for Computing Machinery, New York, NY, USA, 61–72. https://doi.org/10.1145/3198458.3198464

[21] Abdulmotaleb El Saddik. 2018. Digital Twins: The Convergence of Multimedia Technologies. *IEEE MultiMedia* 25 (04 2018), 87–92. https://doi.org/10.1109/MMUL.2018.023121167

[22] J. Fernandez, Pablo Cortés, Jesús Muñuzuri, and J. Guadix. 2014. Dynamic Fuzzy Logic Elevator Group Control System With Relative Waiting Time Consideration. *Industrial Electronics, IEEE Transactions on* 61 (09 2014), 4912–4919. https://doi.org/10.1109/TIE.2013.2289867

[23] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. 2008. Knowledge transfer via multiple model local structure mapping. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 283–291. https://doi.org/10.1145/1401890.1401928

[24] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. 2019. From Variational to Deterministic Autoencoders. https://doi.org/10.48550/ARXIV.1903.12436

[25] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. 2007. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69, 2 (2007), 243–268. https://doi.org/10.1111/j.1467-9868.2007.00587.x

[26] Yoshiyuki Harada, Yoriyuki Yamagata, Osamu Mizuno, and Eun Hye Choi. 2017. Log-based anomaly detection of CPS using a statistical method. *Proceedings - 8th IEEE International Workshop on Empirical Software Engineering in Practice, IWESEP 2017* (2017), 1–6. https://doi.org/10.1109/IWESEP.2017.12 arXiv:1701.03249

[27] Jan Hauke and Tomasz Kossowski. 2011. Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data. *Quaestiones Geographicae* 30 (06 2011), 87–93. https://doi.org/10.2478/v10117-011-0021-1

[28] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. 2017. Cyber-physical systems security—A survey. *IEEE Internet of Things Journal* 4, 6 (2017), 1802–1831. https://doi.org/10.1109/JIOT.2017.2703172

[29] Julian Karch. 2020. Improving on Adjusted R-Squared. *Collabra: Psychology* 6 (09 2020), 45. https://doi.org/10.1525/collabra.343

[30] Angelos Katharopoulos and Francois Fleuret. 2018. Not All Samples Are Created Equal: Deep Learning with Importance Sampling. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 2525–2534. https://proceedings.mlr.press/v80/katharopoulos18a.html

[31] Jay Lee, Moslem Azamfar, Jaskaran Singh, and Shahin Siahpour. 2020. Integration of digital twin and deep learning in cyber-physical systems: towards smart manufacturing. *IET Collaborative Intelligent Manufacturing* 2, 1 (2020), 34–36. https://doi.org/10.1049/iet-cim.2020.0009

[32] Han Liping, Yue Tao, Ali Shaukat, Arrieta Aitor, and Arratibel Maite. 2022. Are Elevator Software Robust Against Uncertainties? Results and Experiences from an Industrial Case Study. In *Proceedings of the 30th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. https://doi.org/10.1145/3540250.355895

[33] Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. 2019. Accurate uncertainty estimation and decomposition in ensemble learning. *advances in neural information processing systems* 32 (2019). https://doi.org/10.48550/arXiv.1911.04061

[34] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*. PMLR, 97–105. https://doi.org/10.48550/arXiv.1502.02791

[35] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. 2015. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems* 80 (2015), 14–23. https://doi.org/10.1016/j.knosys.2015.01.010

[36] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng Daphne Yao. 2020. Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities. 1, 1 (2020), 1–29. https://doi.org/10.1145/3453155 arXiv:2003.13213

[37] Chen Lv, Yang Xing, Junzhi Zhang, Xiaoxiang Na, Yutong Li, Teng Liu, Dongpu Cao, and Fei-Yue Wang. 2017. Levenberg–Marquardt backpropagation training of multilayer neural networks for state estimation of a safety-critical cyber-physical system. *IEEE Transactions on Industrial Informatics* 14, 8 (2017), 3436–3446. https://doi.org/10.1109/TII.2017.2777460

[38] Larry R Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications* 5 (2001), 64–67. https://doi.org/10.1201/9781420049176

[39] Todd A Oliver and Robert D Moser. 2011. Bayesian uncertainty quantification applied to RANS turbulence models. In *Journal of Physics: Conference Series*, Vol. 318. IOP Publishing, 042032. https://doi.org/10.1088/1742-6596/318/4/042032

[40] Seung-Tae Park and Bo-Suk Yang. 2010. An implementation of risk-based inspection for elevator maintenance. *Journal of mechanical science and technology* 24, 12 (2010), 2367–2376. https://doi.org/10.1007/s12206-010-1004-1

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019). https://doi.org/10.48550/arXiv.1912.01703

[42] R. Renner and S. Wolf. 2004. Smooth Renyi entropy and applications. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* 233–. https://doi.org/10.1109/ISIT.2004.1365269

[43] Giedre Sabaliauskaite and Aditya P Mathur. 2015. Aligning cyber-physical system safety and security. In *Complex Systems Design & Management Asia*. Springer, 41–53. https://doi.org/10.1007/978-3-319-12544-2_4

[44] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. 880–887. https://doi.org/10.1145/1390156.1390267

[45] Roman Schefzik, Thordis L Thorarinsdottir, and Tilmann Gneiting. 2013. Uncertainty quantification in complex simulation models using ensemble copula coupling. *Statistical science* 28, 4 (2013), 616–640. https://doi.org/10.1214/13-STS443

[46] Tabea Schmidt, Florian Hauer, and Alexander Pretschner. 2020. Automated Anomaly Detection in CPS Log Files. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 179–194. https://doi.org/0.1007/978-3-

030-54549-9_12
[47] Erwin Schoitsch. 2012. Cyber-Physical Systems (CPS)-What Can We Learn from Disasters with Respect to Assessment, Evaluation and Certification/Qualification of" Systems-of-Systems"?. In *IDIMT*. 69–82.
[48] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306. https://doi.org/10.1016/j.physd.2019.132306
[49] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6, 1 (2019), 1–48. https://doi.org/10.1186/s40537-019-0197-0
[50] Daniel Sonntag, Sonja Zillner, Patrick van der Smagt, and András Lörincz. 2017. Overview of the CPS for smart factories project: Deep learning, knowledge acquisition, anomaly detection and intelligent user interfaces. In *Industrial internet of things*. Springer, 487–504. https://doi.org/10.1007/978-3-319-42559-7_19
[51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
[52] Timothy John Sullivan. 2015. *Introduction to uncertainty quantification*. Vol. 63. Springer. https://doi.org/10.1007/978-3-319-23395-6
[53] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 15, 4 (2019), 2405–2415. https://doi.org/10.1109/TII.2018.2873186
[54] Emre Oner Tartan, Hamit Erdem, and Ali Berkol. 2014. Optimization of waiting and journey time in group elevator system using genetic algorithm. In *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*. IEEE, 361–367. https://doi.org/10.1109/INISTA.2014.6873645
[55] Markus Tauber and Christoph Schmittner. 2018. Enabling security and safety evaluation in industry 4.0 use cases with digital twins. *ERCIM News* (2018).
[56] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014). https://doi.org/10.48550/ARXIV.1412.3474
[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008. https://doi.org/10.48550/ARXIV.1706.03762
[58] Luis Villar-Fidalgo, Adolfo Crespo Márquez, Vicente González-Prida, Antonio De la Fuente, Pablo Martínez-Galán, and Antonio Jesús Guillén López. 2018. Cyber physical systems implementation for asset management improvement: A framework for the transition. *Safety and Reliability–Safe Societies in a Changing World: Proceedings of ESREL 2018, June 17-21, 2018, Trondheim, Norway* (2018). https://doi.org/10.1201/9781351174664-383
[59] Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. 2018. Adversarial distillation of bayesian neural network posteriors. In *International conference on machine learning*. PMLR, 5190–5199.
[60] Tian Wang, Jiakun Li, Yingjun Deng, Chuang Wang, Hichem Snoussi, and Fei Tao. 2021. Digital twin for human-machine interaction with convolutional neural network. *International Journal of Computer Integrated Manufacturing* 34, 7-8 (2021), 888–897. https://doi.org/10.1080/0951192X.2021.1925966
[61] Michael Weiss and Paolo Tonella. 2021. Uncertainty-wizard: Fast and user-friendly neural network uncertainty quantification. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 436–441. https://doi.org/10.1109/ICST49551.2021.00056
[62] Chathurika S Wickramasinghe, Daniel L Marino, Kasun Amarasinghe, and Milos Manic. 2018. Generalization of deep learning for cyber-physical system security: A survey. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 745–751. https://doi.org/10.1109/IECON.2018.8591773
[63] Yan Xu, Yanming Sun, Xiaolong Liu, and Yonghua Zheng. 2019. A digital-twin-assisted fault diagnosis using deep transfer learning. *Ieee Access* 7 (2019), 19990–19999. https://doi.org/10.1109/ACCESS.2018.2890566
[64] Tao Yue, Paolo Arcaini, and Shaukat Ali. 2020. Understanding digital twins for cyber-physical systems: a conceptual model. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 54–71. https://doi.org/10.1007/978-3-030-83723-5_5
[65] Man Zhang, Shaukat Ali, and Tao Yue. 2019. Uncertainty-wise test case generation and minimization for cyber-physical systems. *Journal of Systems and Software* 153 (2019), 1–21. https://doi.org/10.1016/j.jss.2019.03.011
[66] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. 2019. Uncertainty-wise cyber-physical system test modeling. *Software & Systems Modeling* 18, 2 (2019), 1379–1418. https://doi.org/10.1007/s10270-017-0609-6
[67] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. 2016. Understanding uncertainty in cyber-physical systems: a conceptual model. In *European conference on modelling foundations and applications*. Springer, 247–264. https://doi.org/10.1007/978-3-319-42061-5_16
[68] Man Zhang, Tao Yue, Shaukat Ali, Bran Selic, Oscar Okariz, Roland Norgre, and Karmele Intxausti. 2018. Specifying uncertainty in use case models. *Journal of Systems and Software* 144 (2018), 573–603. https://doi.org/10.1016/j.jss.2018.06.075
[69] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2017. Supervised representation learning with double encoding-layer autoencoder for transfer learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 2 (2017), 1–17.