Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
DOI: https://doi.org/10.51903/jtie.v4i2.501

Digital-Twin Dispatching for Urban Mobility via Spatio-Temporal Transformers and Offline Reinforcement Learning

Long Zhang*[1], Ruiyan Ma[2], Peter Greg[3]
Email: longz@smu.edu
[1]*Transportation Systems Engineering, Southern Methodist University, TX, USA*
[2]*Software Engineering, UC Irvine, CA, USA*
[2]*Information Technology, Illinois Tech, IL, USA*
*Corresponding Author

Abstract
This study addresses the challenge of optimizing ride-hailing dispatch and repositioning under data limitations by proposing an end-to-end digital-twin dispatching framework that integrates spatio-temporal demand forecasting with offline reinforcement learning. Using publicly available NYC FOIL ride-hailing data aggregated at the dispatching-base level, the research aims to evaluate whether coarse-grained data can still support reliable, reproducible decision-making pipelines. The methodology consists of two main components: (i) multivariate time-series forecasting using baseline models, a temporal convolutional network (TCN), and a spatio-temporal transformer to predict next-day demand; and (ii) a digital-twin simulation combined with an action-constrained offline reinforcement learning approach, including behavior cloning (BC) and Conservative Q-Learning (CQL), to optimize fleet repositioning decisions. Experimental results show that the TCN achieves the best forecasting accuracy on the test period, although dominant demand regions largely drive performance gains. In the control phase, conservative policies such as CQL demonstrate stable performance with reduced repositioning costs, but do not significantly outperform behavior cloning due to limited training data. The findings indicate that, in coarse aggregate settings, operational improvements are more influenced by controlling policy sensitivity than by marginal forecasting gains. This study contributes a reproducible benchmark pipeline and highlights the importance of conservative control strategies, transparent assumptions, and sensitivity analysis when deploying AI-driven mobility systems based on limited or aggregated data.

Keywords: *Digital Twin, Urban Mobility, Ride-Hailing Dispatch, Demand Forecasting, Spatio-Temporal Transformer.*

I. INTRODUCTION

Ride-hailing platforms operate as large, stochastic service systems (Agustin & Rahayu, 2025; Musrifah & Hasanah, 2025; Rokhman et al., 2025). At any moment, riders request trips with uncertain origins, destinations, and timing, while drivers decide when and where to be active. The platform's operational objective is multi-dimensional: it must maximize service quality and marketplace efficiency (high acceptance rate, low waiting time, high driver utilization), control operational costs (incentives, empty mileage, congestion externalities), and meet regulatory and fairness constraints. Dispatching and repositioning sit at the center of this objective. Dispatching assigns incoming requests to available drivers, while proactively repositioning (or incentivizing) idle supply to locations where future demand is expected to materialize. In practice, these two problems are coupled: a decision that improves the next hour's acceptance can create supply shortages later, and aggressive repositioning can increase deadhead driving and reduce driver satisfaction.

Received on May 2025; Revised on Juni, 2025; Accepted on July 2025; Published on August 2025

Data-driven methods have improved both prediction and control in mobility systems. Short-horizon demand forecasts are routinely used for surge pricing, driver incentives, and staffing. At the same time, reinforcement learning (RL) has been proposed for dynamic dispatch and repositioning because it can directly optimize long-term objectives under uncertainty. However, deploying RL in real ride-hailing marketplaces is difficult. Online RL requires exploratory actions that can harm customers or drivers, and production experimentation can be expensive and risky. Even when historical logs are available, naive offline training can lead to value overestimation and out-of-distribution actions, producing brittle policies that perform well in training but fail in deployment (Levine et al., 2020). This motivates conservative offline RL approaches that explicitly account for distribution shift (Fujimoto et al., 2019; Kumar et al., 2020) and evaluation approaches that avoid uncontrolled real-world experimentation.

A digital twin is a virtual representation of a physical system that is continuously informed by data and can be used to test interventions, optimize operations, and plan scenarios (Grieves, 2014; Boschert & Rosen, 2016). For cities and mobility systems, digital twins are particularly appealing because they allow policy evaluation under controlled assumptions, support reproducibility, and make explicit the mapping from raw observational data to decision outcomes (Batty, 2018; Fuller et al., 2020). The key challenge is fidelity: realistic mobility simulators require detailed trip-level traces, travel-time models, and driver behavior. Public data often provide only aggregated views. Consequently, practical digital twins frequently use calibrated surrogates (e.g., capacity and queue approximations) that preserve relative comparisons between policies even when absolute metrics are approximate.

In this study, we develop a surrogate digital twin loop for ride-hailing dispatching using a public dataset. FiveThirtyEight released a set of NYC TLC FOIL response tables describing daily Uber activity by dispatching base. The file Uber-Jan-Feb-FOIL.csv contains, for each day from January 1 to February 28, 2015, the number of active vehicles and completed trips for six dispatching bases (FiveThirtyEight, 2015). Although this dataset does not include trip origins/destinations, travel times, or wait times, it is sufficient for studying base-level demand-supply dynamics, cross-base correlations, and the operational effects of changing the spatial distribution of supply. We therefore position the study as a FOIL-scale aggregate benchmark for reproducible end-to-end comparison, rather than as evidence of production-ready city-wide dispatching. Following the project constraint, we retain base-level spatial granularity to enable a transparent end-to-end pipeline.

In many operational settings, teams begin with coarse aggregates before investing in high-resolution telemetry. Daily base-level demand is sufficient to address several strategic questions:

how stable are spatial demand shares over a week, how strongly do bases co-vary, and how much operational benefit can be gained from coarse-level supply rebalancing. These questions matter for strategic staffing, driver onboarding, and incentive budget planning. At the same time, coarse daily aggregates naturally damp fast within-day dynamics, so they are more suitable for benchmark-style comparison than for claims about operational dispatch superiority. The key is to treat the digital twin as a controlled comparator and to clearly communicate the assumptions under which results hold.

We design a two-part pipeline (Figure 1). First, we train a spatio-temporal forecaster to predict next-day trips per base. We compare a temporal convolutional network (TCN) with a two-stage spatio-temporal transformer that uses temporal self-attention within each base and spatial self-attention across bases (Vaswani et al., 2017; Bai et al., 2018; Xu et al., 2020). Second, we define a daily repositioning Markov decision process (MDP) whose state includes current fleet distribution and predicted demand, whose actions are constrained to data-driven fleet distributions obtained by clustering historical supply patterns (MacQueen, 1967), and whose reward trades off service rate, waiting time, repositioning distance, and imbalance. We then train offline RL controllers using behavior cloning (BC) and Conservative Q-Learning (CQL), and compare them to heuristic baselines. In all cases, evaluation is performed in a digital twin that converts supply and demand to service outcomes via calibrated surrogates, producing operational metrics aligned with marketplace KPIs.

The paper makes five concrete benchmark-oriented contributions: (1) a fully specified preprocessing and chronological split protocol for Uber-Jan-Feb-FOIL.csv; (2) a controlled comparison of baselines, TCN, and a spatio-temporal transformer for one-day-ahead multivariate demand forecasting at the dispatching-base level; (3) a transparent base-level digital twin that defines acceptance, waiting-time proxy, deadhead-mile index, and inequality metrics from FOIL aggregates; (4) an offline RL formulation with an action-constrained, data-driven repositioning action set and implementations of BC, DQN-style fitted Q-learning, and CQL; and (5) end-to-end evaluation tables/figures including ablations and sensitivity analyses that demonstrate how forecast quality and controller conservatism affect dispatch outcomes. The complete pipeline is reproducible from the stated file and deterministic hyperparameters, enabling extensions to richer datasets while keeping claims aligned to a public-data benchmark.

Many published mobility dispatch studies rely on proprietary simulators or data streams, making it difficult to reproduce end-to-end results. Our goal is not to claim state-of-the-art dispatch performance, but to provide a transparent benchmark that others can audit and extend. We therefore prioritize: (i) deterministic preprocessing, (ii) compact models that train quickly on the

small dataset, (iii) an action-constrained control formulation that stays close to historical behavior, and (iv) evaluation metrics that can be computed solely from observed trips and active vehicles. These choices trade fidelity for clarity, which is appropriate for a public-data study.

The remainder of the paper is organized as follows. The Literature Review synthesizes prior work on digital twins, spatio-temporal forecasting, and offline RL for mobility. Research Methods details the dataset, forecasting models, digital twin, offline RL formulation, and evaluation protocol. The Results and Findings report presents forecasting accuracy and policy trade-offs, including flow-level interpretations and sensitivity analyses. The conclusion and recommendation summarize lessons learned and provide practical guidance for extending the approach to higher-resolution data and production settings.

II. LITERATURE REVIEW

The dispatching pipeline studied here sits at the intersection of digital twin systems, spatio-temporal forecasting, and reinforcement learning for mobility. We review each area with emphasis on concepts that directly inform our design choices and evaluation methodology.

A. Digital Twins for Engineered and Urban Systems

The digital twin concept was popularized in engineering contexts as a "virtual replica" of a physical asset, linked via data to enable monitoring, prediction, and optimization (Grieves, 2014). Boschert and Rosen (2016) emphasize the simulation aspect: a twin is not only a static model but a continuously updated simulation that supports what-if analysis. In manufacturing, research on digital twins has grown rapidly; Kritzinger et al. (2018) provide a comprehensive review and propose taxonomies that distinguish among digital models, digital shadows, and full digital twins based on the degree of data coupling. Fuller et al. (2020) identify enabling technologies (sensing, connectivity, analytics) and open challenges, including model management, interoperability, and trustworthiness. Although our application is mobility rather than manufacturing, the same system-level themes apply: a digital twin must define how data updates the model, which aspects of the physical system are represented, and what performance metrics are valid outputs.

Urban digital twins extend these ideas to city-scale infrastructure and human behavior. Batty (2018) discusses digital twins as a shift in urban analytics toward real-time, data-linked representations of cities. In transportation, a twin may incorporate network, demand, and traffic dynamics models, but human behavior and policy constraints often require hybrid modeling. Tao et al. (2018) frame digital twin systems as closed-loop structures that integrate data, models, and services, motivating our integration of forecasting (model) and dispatching (service) within a single evaluation loop. A practical implication is that a mobility digital twin must explicitly state

simplifying assumptions; otherwise, improvements may be artifacts of the simulator rather than robust operational gains. Recent transportation-specific digital-twin work likewise highlights the importance of continuous calibration and explicit data–simulation coupling (Kušić et al., 2023).

B. Spatio-Temporal Forecasting for Mobility Demand

Demand forecasting in mobility is a multivariate time-series problem with spatial dependencies. Traffic forecasting literature often models sensor networks or regions connected by a road graph. Graph neural networks and diffusion processes have been used to model spatial propagation of congestion and flow. Li et al. (2018) propose diffusion convolutional recurrent neural networks (DCRNN) that combine graph diffusion with RNN dynamics, while Yu et al. (2018) introduce spatio-temporal graph convolutional networks (STGCN) that apply graph convolutions and temporal convolutions. For ride-hailing demand specifically, Ke et al. (2017) show that jointly modeling spatial and temporal structure improves short-term passenger-demand forecasting. Even though our dataset has only six bases and no explicit adjacency graph, the general insight still holds: demand at one location predicts demand at others, so joint modeling can improve forecasts.

C. Transformer and Convolutional Approaches for Time Series

Transformer self-attention (Vaswani et al., 2017) has been adapted to time series because it can attend to relevant history without fixed receptive fields, and can model cross-series interactions via attention weights. Lim et al. (2021) propose the Temporal Fusion Transformer, combining attention with gating mechanisms and interpretable variable selection for multi-horizon forecasting. For traffic flow, Xu et al. (2020) propose a spatial-temporal transformer network that explicitly applies attention across time and space. Compared to transformers, temporal convolutional networks provide a strong baseline with fewer parameters and stable training, using dilations to capture longer temporal dependencies (Bai et al., 2018). Our forecasting study uses these insights to build a compact two-stage attention model and a compact TCN, and evaluates both against classical baselines.

D. Reinforcement Learning for Ride-Hailing Dispatch and Fleet Management

Dispatching and repositioning are naturally sequential decision problems. Optimization-based work has studied dynamic assignment and ride-sharing using combinatorial methods; for example, Alonso-Mora et al. (2017) propose a dynamic trip-vehicle assignment algorithm for high-capacity ride-sharing. Learning-based approaches have scaled RL to large fleets by using approximate value functions, multi-agent decompositions, and simulation environments. Lin et al. (2018) present a multi-agent deep RL approach for large-scale fleet management, showing that coordinated policies can outperform heuristics. Zhou et al. (2019) formulate order dispatching as

a distribution-matching problem between vehicles and orders, enabling scalable multi-agent RL training. Holler et al. (2019) directly address dispatching and repositioning as a joint deep RL problem. These studies demonstrate potential gains but typically rely on simulators or online learning, which can be difficult to reproduce without proprietary data. A more deployment-oriented example is Jiao et al. (2021), who combine batch training with decision-time planning for real-world ride-hailing vehicle repositioning. Qin et al. (2022) survey these directions and note the importance of realistic constraints, data support, and evaluation fidelity in ridesharing RL.

E. Offline Reinforcement Learning

Offline RL trains policies on fixed datasets without additional interaction with the environment. Levine et al. (2020) review offline RL and emphasize distribution shift as the central obstacle. If a learned policy selects actions outside the support of the behavior data, value estimates may be unreliable. Batch-constrained approaches explicitly restrict actions to those likely to be present in the dataset; BCQ (Fujimoto et al., 2019) uses a generative model to propose candidate actions. Conservative Q-Learning (Kumar et al., 2020) instead regularizes Q-values by penalizing high values for actions not supported by the dataset, improving robustness in many benchmarks. Implicit Q-Learning (Kostrikov et al., 2021) avoids out-of-distribution max operations by using implicit value estimation. In mobility, offline RL is attractive because historical logs are abundant and online exploration is costly. Our work operationalizes this idea on a public dataset by constraining the action space to data-derived fleet distributions and by using CQL to further reduce overestimation.

F. Evaluation Metrics and Equity

Mobility platforms often track acceptance rate and waiting time as primary customer-facing metrics, while empty mileage and driver utilization capture operational efficiency. Spatial inequity can be quantified by dispersion metrics, including variance of service rates and inequality indices such as the Gini coefficient (Gini, 1921). Because our dataset lacks customer-level wait times, we adopt a queue-inspired waiting-time proxy derived from load factor; such proxies are common in operations research when detailed arrival/service processes are unobserved (Kleinrock, 1975). We treat these metrics as comparative: the digital twin is designed to preserve relative policy ranking under consistent assumptions.

G. Synthesis

Prior work suggests that effective dispatching pipelines should integrate prediction and control, restrict or regularize RL policies to remain within the support of the available data, and report

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

multiple metrics that capture both efficiency and equity. The remainder of the paper implements this synthesis on FOIL-scale aggregates, prioritizing reproducibility and transparency over high-fidelity microscopic simulation.

H. Simulator-Based Evaluation and Surrogate Modeling

Beyond full digital twins, mobility research often uses simulators that approximate demand arrivals, travel times, and driver movements. When trip-level traces are unavailable, surrogate models are a practical alternative. The operations research literature has long relied on queueing approximations to estimate delays and service levels in systems with stochastic arrivals and limited service capacity (Kleinrock, 1975). In machine learning, surrogate simulators enable counterfactual evaluation of policies without risky deployment, but the validity of conclusions depends on how well the surrogate preserves policy ranking. Our work positions the base-level twin as such a surrogate: it is not intended to reproduce exact waits or routes, but it allows systematic comparisons under fixed assumptions. It highlights interactions between prediction error and control sensitivity.

III. RESEARCH METHOD

The overall workflow of the proposed study integrates demand forecasting, digital twin simulation, and offline reinforcement learning for fleet dispatch optimization. Historical FOIL data are first used to train spatio-temporal forecasting models that predict next-day demand across dispatching bases. These predictions are then fed into a digital-twin environment that approximates queue dynamics and vehicle capacity constraints. Finally, offline reinforcement learning algorithms learn repositioning policies from historical transitions under a constrained action space. The complete end-to-end pipeline used in this study is illustrated in Figure 1.

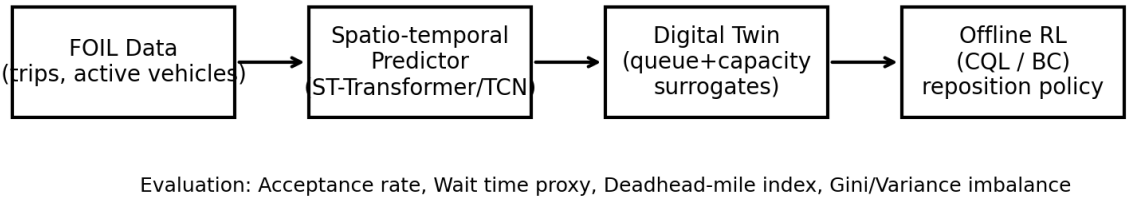

Figure 1. End-to-End Digital-Twin Dispatching Pipeline Evaluated in This Study

This section describes the dataset, preprocessing, forecasting models, digital twin design, offline RL formulation, and evaluation protocol. We explicitly separate (a) what is directly observed in the FOIL file and (b) what is assumed in the digital twin. All experiments were conducted on Uber-Jan-Feb-FOIL.csv using deterministic seeds and fixed hyperparameters, as documented in the tables.

A. Data Representation

Let $t$denote the time index (days) and $i$denote the dispatching-base index. The dataset provides two daily variables: demand $D_{t,i}$, representing the number of trips, and supply $V_{t,i}$, representing the number of active vehicles. The long-form data are reshaped into matrix representations $\mathbf{D} \in \mathbb{R}^{T\times N}$and $\mathbf{V} \in \mathbb{R}^{T\times N}$, where $T = 59$denotes the number of days and $N = 6$denotes the number of dispatching bases. The total daily demand and supply are defined as $D_t^{\text{tot}} = \sum_{i=1}^{N} D_{t,i}$and $V_t^{\text{tot}} = \sum_{i=1}^{N} V_{t,i}$, respectively. Furthermore, the fleet proportion at each base is defined as $p_{t,i} = \frac{V_{t,i}}{V_t^{\text{tot}}}$, which serves as the controllable decision variable in the repositioning Markov Decision Process (MDP)

B. Chronological Splits

Because the objective is to support operational forecasting and control, a strictly chronological data split is employed instead of random shuffling. The training period spans January 1 to February 14, 2015 (45 days), the validation period covers February 15 to February 21, 2015 (7 days), and the testing period includes February 22 to February 28, 2015 (7 days). For the forecasting task, a lookback window of $L = 14$days is used. Each supervised instance predicts the next-day demand vector $\mathrm{D}_t$based on historical observations $\{\mathrm{D}_{t-L}, \dots, \mathrm{D}_{t-1}\}$, $\{\mathrm{V}_{t-L}, \dots, \mathrm{V}_{t-1}\}$, and the day-of-week feature corresponding to time t. This configuration results in 31 training samples, 7 validation samples, and 7 testing samples.

C. Handling Scale and Normalization

Trips and active vehicles have different magnitudes and base-specific scales. For neural forecasting models, we standardize the flattened input tensor over the training samples using a single StandardScaler, which preserves relative variations across bases while improving optimization stability. Targets (next-day trips) are standardized per base using training statistics. We then invert the standardization for all reported metrics so results are in the original trip units. Summary statistics of the dataset and the experimental data splits are presented in Tables 1, 2, and 3.

Table 1. Dataset Summary for Uber-Jan-Feb-FOIL.csv (Daily Aggregates by Dispatching Base)

Statistic	Value
NumDays	59
NumBases	6
StartDate	2015-01-01
EndDate	2015-02-28
TotalTrips	4130230
TotalActiveVehicles	462832
AvgTripsPerDay	70003.898
AvgVehiclesPerDay	7844.610

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Table 2. Chronological Data Split and Number of Supervised Forecasting Samples (lookback L=14)

Split	StartDate	EndDate	NumDays	NumSamples(L=14)
Train	2015-01-01	2015-02-14	45	31
Validation	2015-02-15	2015-02-21	7	7
Test	2015-02-22	2015-02-28	7	7

Table 3. Base-Level Descriptive Statistics (Jan 1–Feb 28, 2015)

Base	TotalTrips	MeanTrips	StdTrips	TotalVehicles	MeanVehicles	StdVehicles	MeanTripsPerVehicle
B02512	93786	1589.593	384.228	13125	222.458	33.423	7.073
B02598	540791	9165.949	2060.766	58653	994.119	134.304	9.134
B02617	725025	12288.559	2529.838	79758	1351.831	161.360	9.024
B02682	662509	11228.966	2838.999	71431	1210.695	190.818	9.161
B02764	1914449	32448.288	6788.452	217290	3682.881	438.326	8.754
B02765	193670	3282.542	2019.321	22575	382.627	180.135	8.105

D. Forecasting Models

We evaluate five forecasting models. (i) The weekday-mean model predicts demand at each base using the historical mean corresponding to the same weekday, thereby capturing weekly seasonality. (ii) The persistence model assumes that the next-day demand equals the current day, which serves as a strong baseline for short-horizon time series. (iii) Ridge regression employs a linear mapping from lagged demand and supply variables to next-day demand, with ℓ_2-regularization to reduce overfitting.

Table 4. Demand Forecasting Model Configurations Used in the Experiments

Model	Architecture	Training Setup
WeekdayMean	None	Training weekday averages
Persistence	None	y_{t+1}=y_t
RidgeLag	alpha=1.0	Flattened 14-day trips+vehicles + DOW one-hot
TCN	2 conv layers, channels=8, dilation=[1,2], epochs=120 (select 65)	Adam lr=5e-3, wd=1e-4, dropout=0.1
ST-Transformer	d_model=16, heads=2, 1 temporal + 1 spatial layer, epochs=160 (select 40)	Adam lr=3e-3, wd=1e-4, dropout=0.1

(iv) The temporal convolutional network (TCN) treats concatenated base-level features as input channels and applies dilated one-dimensional convolutions over the L-day history. Formally, given an input tensor $\mathrm{X} \in \mathbb{R}^{L\times(N\cdot F)}$, where $F = 2$denotes the number of features per base, a TCN layer computes $\mathrm{H} = \sigma(\mathrm{Conv1D}_{\mathrm{dilated}}(\mathrm{X}))$, enabling an expanded receptive field beyond the kernel size. (v) The spatio-temporal transformer adopts a two-stage attention mechanism consisting of temporal attention within each base and spatial attention across bases. Given embedded inputs $\mathrm{e}_{t,i} \in \mathbb{R}^{d}$, temporal representations are first obtained as $\mathrm{h}_i^{\mathrm{time}} = \mathrm{Transformer}_{\mathrm{time}}(\mathrm{e}_{:,i})$for each base i. These are then processed through spatial attention as $\mathrm{h}_t^{\mathrm{space}} = \mathrm{Transformer}_{\mathrm{space}}(\mathrm{h}_{t,:}^{\mathrm{time}})$, and

the final prediction D_{t+1} is generated from the representation at the last time step. This architecture follows prior spatio-temporal transformer designs in traffic forecasting while remaining compact and suitable for FOIL-scale datasets (Xu et al., 2020). The detailed architectural configurations and training settings for all models are summarized in Table 4.

E. Training and Evaluation of Forecasters

For neural models, we standardize inputs and targets using training statistics, optimize for mean-squared error, and select the best checkpoint based on validation MAE. We report MAE and RMSE in trips as primary metrics. We also compute MAPE and sMAPE for completeness; however, percentage metrics can be unstable when true values are small. Because N=6 is small, we evaluate both overall errors (aggregated across bases) and per-base errors to identify uneven predictive performance that could propagate into control decisions. Because the validation and test windows each contain only seven days, we interpret model differences descriptively and supplement overall metrics with per-base and macro-versus-micro comparisons rather than strong significance claims.

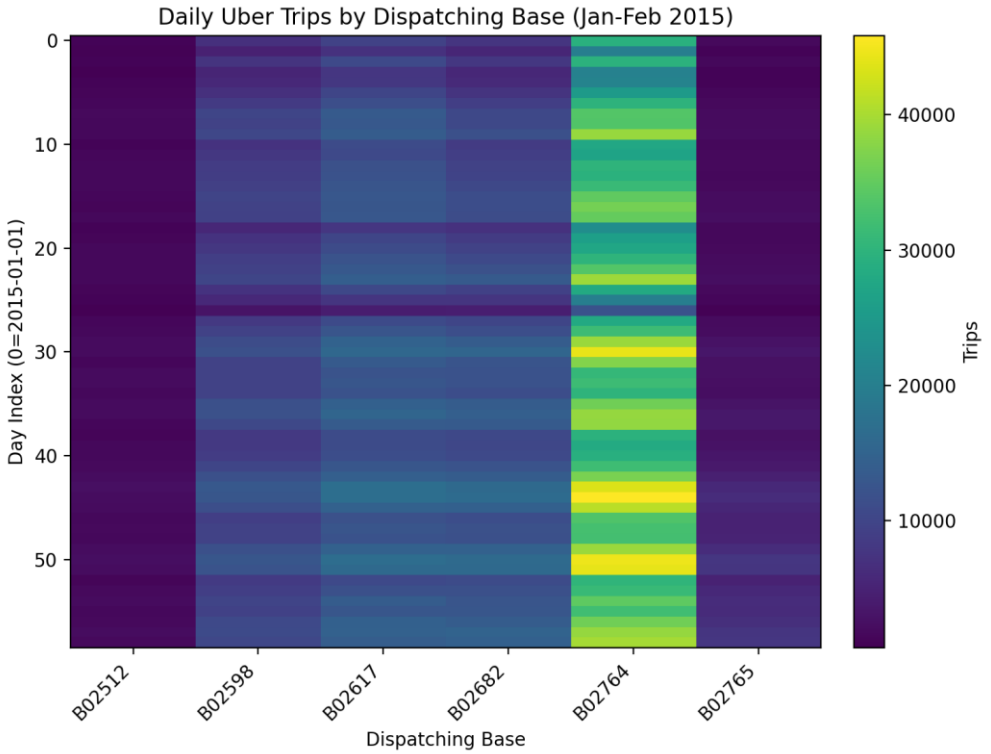

Figure 2. Daily Trips Across Six Dispatching Bases in the FOIL Dataset (Higher Intensity Indicates Higher Demand)

F. Model Capacity and Parameter Efficiency

Because the dataset is small (31 training targets), we intentionally use compact neural architectures. The TCN used in our experiments has 1216 trainable parameters, while the spatio-

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

temporal transformer has 5481 trainable parameters. These counts are orders of magnitude smaller than production-scale forecasting models, and they help mitigate overfitting. In larger datasets, one would typically scale d_model and the number of layers and use regularization strategies such as dropout, early stopping, and possibly data augmentation via calendar features or external covariates. To provide an overview of demand patterns in the dataset, Figure 2 shows daily trip intensity across the six dispatching bases.

G. Digital Twin Design

The FOIL dataset does not provide ride-level service or waiting-time information. To enable policy evaluation, we construct a surrogate mechanism that maps daily supply and demand into served trips and delay estimates. We assume that each active vehicle can serve $c$trips per day, where $c$is calibrated from the training data as the median of the ratio $D_{t,i}/V_{t,i}$across all base-days. This yields $c = 8.38$with an interquartile range of 7.19–9.22. The central bases exhibit similar medians (8.50–8.82), whereas B02512 and B02765 are lower (6.79 and 7.55), indicating that the pooled $c$should be interpreted as a compact global surrogate rather than a base-invariant physical parameter. A base-specific parameter $c_i$may be introduced in larger datasets; however, we retain a single pooled value to avoid over-parameterization in the limited 45-day training period.

Given demand vector dand supply vector v, the served trips at each base are computed as $s_i = \min(d_i, c \cdot v_i)$, and unmet demand is $u_i = d_i - s_i$. The system-level acceptance rate is defined as $\frac{\sum_i s_i}{\sum_i d_i}$. To approximate waiting time, we define a load factor $\rho_i = \frac{d_i}{c \cdot v_i}$and introduce a piecewise-linear waiting-time proxy $w_i = w_{\min} + w_{\text{slope}} \cdot \max(0, \rho_i - 1)$, where $w_{\min} = 3$minutes and $w_{\text{slope}} = 12$minutes. The daily average waiting-time proxy is computed as a demand-weighted mean $w_{\text{avg}} = \frac{\sum_i w_i \cdot d_i}{\sum_i d_i}$. This formulation is motivated by queueing intuition: delays increase when demand exceeds service capacity ($\rho_i > 1$), while remaining near a baseline when capacity is sufficient ($\rho_i \leq 1$). We emphasize that this waiting-time measure serves as a comparative proxy rather than a calibrated estimate of actual passenger delay.

H. Imbalance Metrics

To quantify spatial inequity in service, we compute two imbalance measures daily. First, we calculate the Gini coefficient of unmet demand across bases, denoted as $G(\mathrm{u})$, where higher values indicate that unmet demand is more concentrated in a smaller number of bases (Gini, 1921). Second, we compute the variance of per-base service rates, defined as $\frac{s_i}{d_i}$. In an equitable system, service rates are expected to be relatively uniform across locations, resulting in both low variance and low concentration of unmet demand.

I. Repositioning Cost Proxy

The dataset does not include vehicle trajectory information; therefore, repositioning (deadhead) cost is approximated using changes in fleet distribution. Let $\mathrm{p}_t$denote the fleet proportion vector at the beginning of day t, and p_{t+1}denote the selected distribution for day $t+1$. The number of moved vehicles is defined as

$$\mathrm{moved}_t = 0.5 \cdot \min\left(V_t^{\mathrm{tot}}, V_{t+1}^{\mathrm{tot}}\right) \cdot \| \mathrm{p}_{t+1} - \mathrm{p}_t \|_1,$$

which represents the minimum number of vehicles that must be reassigned to transform one distribution into another under a mass-balance assumption.

The deadhead-mile index is then defined as

$$\mathrm{deadhead}_t = \mathrm{moved}_t \cdot m,$$

where $m = 2$miles per moved vehicle. This index serves as a proxy for empty repositioning distance. It is intentionally conservative, as it attributes all changes in fleet distribution to repositioning, whereas in practice such changes may also result from completed trips or driver shift dynamics. Consequently, this metric is interpreted primarily as an indicator of policy aggressiveness capturing the extent of fleet redistribution across bases, rather than as an exact measure of physical travel distance. It does not explicitly model routing, congestion effects, trip-induced flows, or driver behavioral dynamics.

J. Constructing Implied Reposition Flows

For visualization purposes (Figure 6), we construct an implied flow matrix between dispatching bases by comparing consecutive fleet distributions. Both distributions are first projected onto a common total fleet size, $\min(V_t^{\mathrm{tot}}, V_{t+1}^{\mathrm{tot}})$, after which the net change at each base is computed. Bases with negative net changes are treated as sources, while those with positive net changes are treated as sinks. A greedy transport matching procedure is then applied to allocate flows from sources to sinks.

This process produces a non-negative flow matrix whose row sums correspond to total outgoing movements and whose column sums correspond to total incoming movements. Although this formulation does not represent actual vehicle trajectories or routing behavior, it provides a useful abstraction for interpreting how different policies amplify or reduce specific base-to-base reallocations.

K. Offline RL MDP and Action Set

We model a daily decision process at the end of day t, in which a policy selects the fleet distribution for day $t+1$. The state $\mathrm{s}_t$is defined as the concatenation of (i) current fleet

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

proportions p_t, (ii) predicted next-day demand $\hat{\mathrm{d}}_{t+1}$, (iii) the day-of-week encoding for $t+1$, and (iv) the total number of vehicles V_{t+1}^{tot}.

To ensure stability in offline reinforcement learning, the action space is constrained to a discrete, data-driven set. Specifically, historical fleet proportion vectors $\mathrm{p}_t$from the training period are clustered using k-means with $K = 8$clusters (MacQueen, 1967). Each action $a$corresponds to a centroid distribution $\mathrm{p}^{(a)}$. When an action $a$is selected, the digital twin allocates the next-day fleet as

$$\mathrm{v}_{t+1} = \mathrm{p}^{(a)} \cdot V_{t+1}^{\mathrm{tot}}.$$

This action constraint serves two primary purposes: (i) it restricts policies to remain within the support of observed data (which is critical in offline RL settings), and (ii) it provides an interpretable set of operational allocation modes. The resulting centroid distributions and their frequencies in the training data are summarized in Table 5.

Table 5. Data-Driven Discrete Action Set: K-Means Centroid Fleet Proportions and Training Frequency (Days)

Action	B02512	B02598	B02617	B02682	B02764	B02765	TrainFreqDays
A0	0.028929133	0.1306941	0.18178569	0.14436641	0.48019087	0.034033865	10
A1	0.029153379	0.12895702	0.1737945	0.15815817	0.47159338	0.038343553	11
A2	0.029541187	0.12868564	0.17027582	0.16163766	0.45790112	0.05195855	3
A3	0.0270356	0.12711309	0.18075305	0.13912867	0.4942015	0.03176807	5
A4	0.027089758	0.12567961	0.1755774	0.15487641	0.4816882	0.035088614	8
A5	0.028348561	0.12662761	0.16636372	0.15954527	0.4570831	0.062031716	3
A6	0.02825904	0.11875526	0.1668629	0.14852817	0.49756098	0.040033642	1
A7	0.031035319	0.13047127	0.17273435	0.16519944	0.46234626	0.03821336	4

L. Reward Design and Offline Dataset

We define a scalar reward function to train reinforcement learning controllers that captures a multi-objective trade-off. For each day $t+1$, given the selected supply $\mathrm{v}_{t+1}$and realized demand d_{t+1}, we compute the acceptance rate, the waiting-time proxy w_{avg}, the deadhead-mile index between $\mathrm{p}_t$and p_{t+1}, and the Gini coefficient of unmet demand. The reward is then defined as

$$r_t = \text{acceptance} - 0.02 \cdot w_{\mathrm{avg}} - 0.0005 \cdot \text{deadhead} - 0.1 \cdot G(\mathrm{u}).$$

The weighting coefficients are chosen such that each component contributes a comparable magnitude under typical training transitions, thereby preventing any single term from dominating the optimization process. Empirically, the median weighted contributions on the behavior dataset are approximately 0.071 for $0.02 \cdot w_{\mathrm{avg}}$, 0.062 for $0.0005 \cdot$ deadhead, and 0.044 for $0.1 \cdot G(\mathrm{u})$, compared to a median acceptance term of 0.957. Nevertheless, these weights reflect a design preference over service quality, operational cost, and equity, rather than a uniquely defined ground-truth objective.

Offline transitions $(s_t, a_t, r_t, s_{t+1})$are constructed from the training period by assigning the behavior action as the cluster index corresponding to the observed next-day fleet distribution. This results in a total of 31 transitions, which is relatively small. Accordingly, the offline RL analysis should be interpreted as a feasibility study under conservative action constraints, rather than as a data-rich benchmark. The detailed configuration of the offline RL algorithms and the digital-twin evaluation environment is provided in Table 6.

Table 6. Offline RL and Digital-Twin Configuration Used for Dispatching Evaluation

Component	Setting	Notes
Action set	K=8 k-means centroids	Clusters on training-day supply proportions
State	p_t (6) + predicted demand (6) + DOW one-hot (7) + V_{t+1} (1)	Total dimension 20
BC	2-layer MLP (64-64), 500 epochs	Cross-entropy, Adam lr=1e-3
DQN	2-layer MLP (64-64), 1500 iterations	gamma=0.95, soft target tau=0.01
CQL	Same as DQN with conservative term	alpha=1.0, gamma=0.95, tau=0.01, 1500 iterations
Deadhead index	2 miles per moved vehicle	Moved vehicles = 0.5*min(V_t,V_{t+1})*‖p_{t+1}-p_t‖_1

M. Offline RL Algorithms

Behavior cloning (BC) learns a policy $\pi(a \mid \mathbf{s})$through supervised learning by imitating observed behavior actions. Fitted Q-learning (DQN-style) estimates a state–action value function $Q(s, a)$using temporal-difference targets defined as

$$y = r + \gamma \max_{a'} Q(\mathbf{s}', a'),$$

with a slowly updated target network to improve training stability (Mnih et al., 2015).

In offline settings, the maximization operator may evaluate actions not supported by the dataset, leading to value overestimation. Conservative Q-Learning (CQL) mitigates this issue by introducing a regularization term that penalizes overestimated Q-values. In the discrete-action setting considered here, the CQL objective augments the temporal-difference loss with a conservative penalty of the form

$$\alpha \left(\log \sum_{a} \exp\ (Q(\mathbf{s}, a)) - Q(\mathbf{s}, a_{\text{behavior}}) \right),$$

which encourages the estimated Q-values of unseen actions to remain lower than those of observed (behavioral) actions (Kumar et al., 2020). In our implementation, we use a discount factor $\gamma = 0.95$, a conservative regularization coefficient $\alpha = 1.0$, and soft target updates with parameter $\tau = 0.01$.

N. Evaluation Protocol

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

We evaluate policies in a closed-loop simulation during the validation and test weeks. At the start of each evaluation day d, the policy observes the state derived from the previous day's fleet distribution and the forecast for day d. It selects an action (fleet distribution) for day d, the digital twin computes service outcomes using the realized demand d, and the simulation transitions to the next day using the chosen distribution as the new state. We report the total acceptance rate over the evaluation horizon, the average waiting-time proxy, the total deadhead-mile index, and the average imbalance metrics. In addition to learned policies, we evaluate Uniform allocation, NoReposition, and PredDemandHeuristic (allocate proportionally to predicted demand). We also include DemandOracle (allocate proportionally to realized demand) as an upper bound, noting that it is not implementable in practice.

O. Pseudo-Algorithm for Closed-Loop Evaluation

The evaluation loop can be summarized as: (1) initialize the current fleet distribution with the observed distribution on the day before the evaluation horizon; (2) for each evaluation day d, construct state s from current distribution, forecast for day d, calendar features, and total vehicles; (3) select an action (next-day distribution) using the policy; (4) compute service outcomes and costs in the digital twin using realized demand; and (5) update the current distribution to the selected distribution and continue. This closed-loop structure is essential: it captures how a policy's decisions affect future states, rather than evaluating each day independently.

P. Reproducibility Checklist

To reproduce results, one needs only the FOIL file and the fixed split dates. Key reproducibility elements include: fixed random seeds for model initialization and k-means clustering, consistent standardization using training statistics, and deterministic evaluation of metrics. We provide all hyperparameters in Tables 4–6. Because the dataset is small, training and evaluation complete quickly on a CPU, which further supports reproducibility for academic and educational use.

IV. RESULT

We present results for demand forecasting and dispatching policies. All numerical values are computed from the fixed chronological splits described earlier. Figure 3 compares the forecasting performance of the evaluated models on the held-out test week using mean absolute error (MAE) measured in trips. Lower MAE indicates better predictive accuracy. Among the evaluated models, the TCN achieves the lowest error, outperforming both classical baselines and the spatio-temporal transformer, while Ridge regression performs substantially worse due to its limited ability to capture nonlinear temporal patterns.

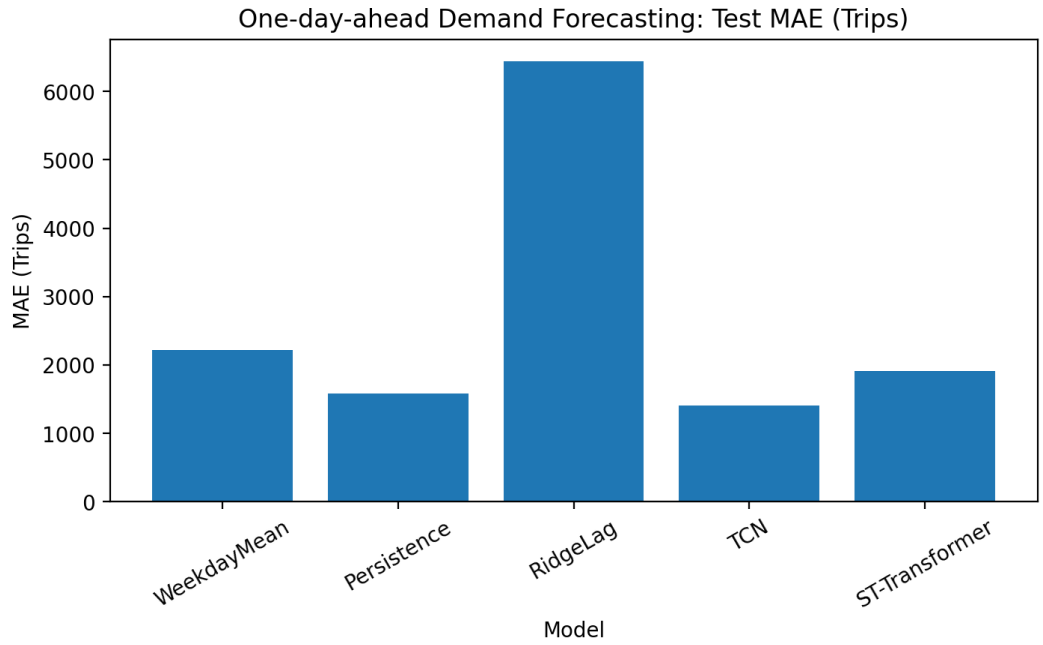

Figure 3. Forecasting Comparison on the Held-Out Test Week Using MAE in Trips (Lower is Better)

To further illustrate model behavior, Figure 4 shows an example of one-day-ahead forecasts during the test week for dispatching base B02764. The plot compares the actual observed trips with predictions from the TCN and spatio-temporal transformer models. Both neural models capture the general upward demand trend toward the end of the week, although the ST-Transformer tends to slightly overestimate demand compared to the TCN predictions.

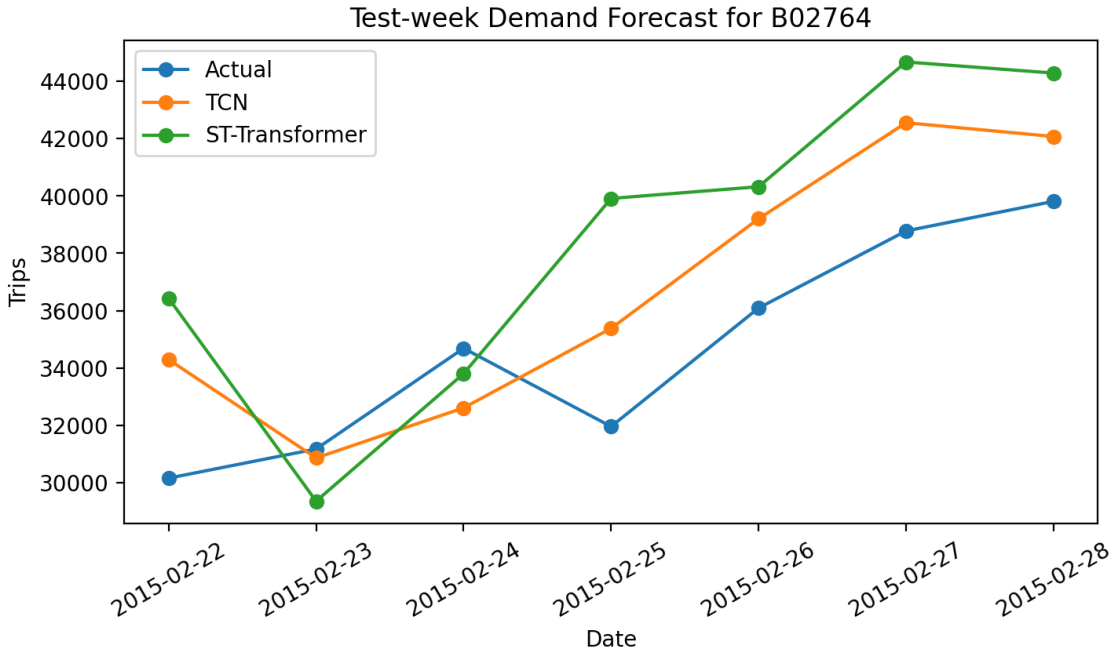

Figure 4. Example One-Day-Ahead Demand Forecasts on the Test Week for Base B02764

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

We present results for demand forecasting and dispatching policies. All numerical values are computed from the fixed splits described earlier.

A. Descriptive Patterns

Table 3 shows that demand and supply are highly concentrated in base B02764, which accounts for the majority of total trips and active vehicles. This concentration implies that overall forecasting errors are largely driven by performance on B02764, while smaller bases can exhibit higher relative volatility. Figure 2 visualizes the daily trip matrix and reveals strong temporal co-movement across bases, suggesting shared city-wide factors. These correlations motivate joint multivariate forecasting rather than independent per-base models.

B. Forecasting Results

Table 7 compares five models. The persistence baseline is strong, reflecting smooth day-to-day demand at the base level. The weekday-mean baseline captures weekly seasonality but cannot adapt to local deviations. On the test week, the TCN achieves the lowest MAE (1406.2) and RMSE (1891.0), outperforming persistence (MAE 1586.1). However, because the evaluation horizon contains only seven days, we interpret the TCN-versus-transformer gap as descriptive rather than statistically decisive. The spatio-temporal transformer achieves lower validation MAE than TCN but higher test MAE, consistent with mild overfitting in a low-data regime. One interpretation is that attention provides flexible modeling capacity, but without sufficient training examples, the model may fit noise in cross-base interactions. In practical deployments with richer data (hourly, spatial grids), transformers often improve; in FOIL-scale data, TCN appears to be a safer benchmark choice.

Table 7. Demand Forecasting Accuracy on Validation and Test Sets (Lower is Better)

Model	Val_MAE	Val_RMSE	Val_MAPE%	Test_MAE	Test_RMSE	Test_MAPE%
Persistence	1514.738	2355.125	11.486	1586.071	2849.081	14.233
WeekdayMean	3219.659	4076.564	27.061	2217.333	2952.342	21.944
RidgeLag	2426.658	3407.651	21.833	6436.202	7978.863	59.224
TCN	1301.990	1933.840	13.295	1406.157	1890.980	14.080
ST-Transformer	945.495	1364.622	9.530	1909.808	2531.902	16.535

C. Per-Base Errors and Operational Relevance

Table 8 shows that model performance differs across bases. For B02764, both TCN and the transformer reduce MAE relative to persistence, which is operationally important because B02764 accounts for the majority of demand. However, for B02765 the TCN error is higher than persistence, indicating that the model may underfit or overfit minority patterns. This matters for dispatching: if a controller overreacts to noisy predictions in small bases, it can induce

unnecessary repositioning and increase inequality. This motivates conservative control mechanisms and action constraints.

Table 8. Per-Base Test MAE (trips) for Representative Forecasting Models

Base	Persistence	TCN	ST-Transformer
B02512	325.42856	136.6265	299.45465
B02598	1125.5714	961.25305	1549.8494
B02617	1693.4286	1075.3152	1973.841
B02682	1383.8572	463.1186	1390.6466
B02764	4164.2856	2728.8022	4505.056
B02765	823.8571	3071.828	1740.0001

A simple macro-versus-micro check clarifies how strongly B02764 influences aggregate results. Averaging per-base test MAE equally across all six bases reproduces the overall ranking in Table 7 (TCN 1406.2, persistence 1586.1, transformer 1909.8). However, excluding B02764, persistence slightly outperforms TCN (1070.4 vs. 1141.6), while the transformer remains highest at 1390.8. Thus, much of the TCN's aggregate advantage is driven by the dominant base, whereas smaller-base performance is more mixed—especially for B02765. We therefore treat the forecasting comparison as a macro-base benchmark result, not as uniform superiority across all bases.

D. Error Structure and Small-Data Behavior

A notable pattern in Table 7 is that the ridge regression model performs much worse than simpler baselines on the test week. With only 31 training targets but 175 input features (14 days × 6 bases × 2 signals, plus 7 day-of-week indicators), the linear model operates in a high-dimensional regime where coefficients are sensitive to scaling and week-specific correlations even under L2 regularization. The poor test performance is therefore consistent with overfitting and instability rather than with a structural failure of linear forecasting per se. The same risk applies to neural models: without careful early stopping and compact architectures, models can fit idiosyncratic fluctuations that do not generalize. This observation reinforces the importance of the validation week and motivates conservative controllers downstream.

E. Policy Comparison

Table 9 and Figure 5 summarize dispatch/repositioning performance on the test week. The behavior baseline (observed fleet proportions) yields high acceptance and a moderate waiting-time proxy. Uniform allocation performs worst because it ignores demand concentration; this is expected given that demand is not spatially uniform. NoReposition performs surprisingly well in this dataset because, at daily base-level aggregates, demand-share vectors are relatively stable from one day to the next. This is an important empirical finding: in such a coarse regime, performance gains arise less from complex dynamic control and more from avoiding overreaction

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

to modest forecast fluctuations. Accordingly, the value of RL in the present setting is primarily stabilization rather than a large improvement in service rate.

Table 9. Digital-Twin Policy Evaluation on the Test Week (Feb 22–28, 2015)

Policy	AcceptanceRate	AvgWait (min)	DeadheadMiles	MovedVehicles	GiniUnmet	VarServiceRate
Behavior	0.9184	4.119	853.461	426.731	0.3987	0.0020
NoReposition	0.9178	4.138	0.0000	0.0000	0.3918	0.0026
Uniform	0.6903	13.186	4381.899	2190.949	0.8009	0.0550
PredDemandHeuristic	0.9190	4.277	1499.691	749.845	0.5110	0.0144
BC	0.9146	4.329	577.076	288.538	0.5112	0.0156
CQL	0.9146	4.329	577.076	288.538	0.5112	0.0156
DemandOracle	0.9236	4.035	1566.719	783.360	0.3976	0.0000
DQN	0.9141	4.418	399.184	199.592	0.5360	0.0198

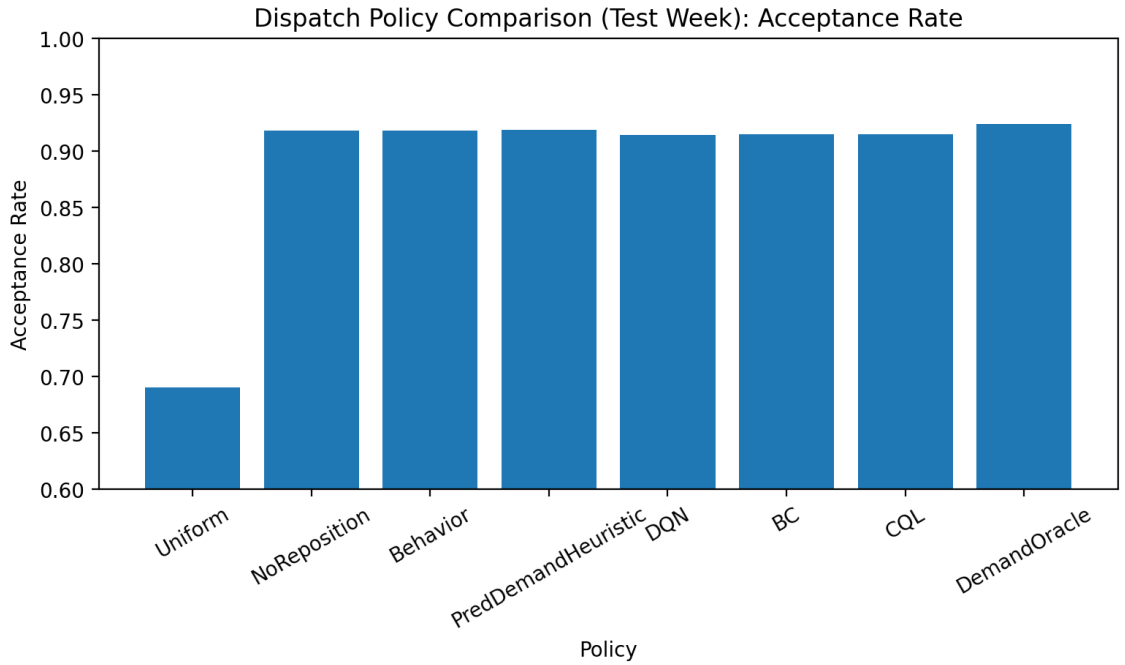

Figure 5. Acceptance Rate Across Dispatch/Repositioning Policies on the Test Week

PredDemandHeuristic reallocates supply proportionally to predicted demand each day. This improves neither acceptance nor waiting time relative to behavior, but greatly increases the deadhead-mile index because it moves fleet mass frequently. This illustrates a key systems insight: even reasonably accurate predictions can be operationally counterproductive if the control law is overly sensitive. The DemandOracle provides an upper bound on acceptance because it uses realized demand; however, it also induces large rebalancing cost, highlighting that aggressive reallocation is attractive only when deadhead is weakly penalized and that policy ranking depends on the chosen multi-objective trade-off.

F. Offline RL Controllers

BC and CQL achieve the same closed-loop metrics in this small action-constrained setting. Given only 31 offline transitions, K=8 discrete actions, and behavior labels concentrated on a few historically common allocation modes, CQL has limited room to separate from imitation. The learned policies therefore select among historically observed fleet allocation modes rather than constructing materially new distributions.

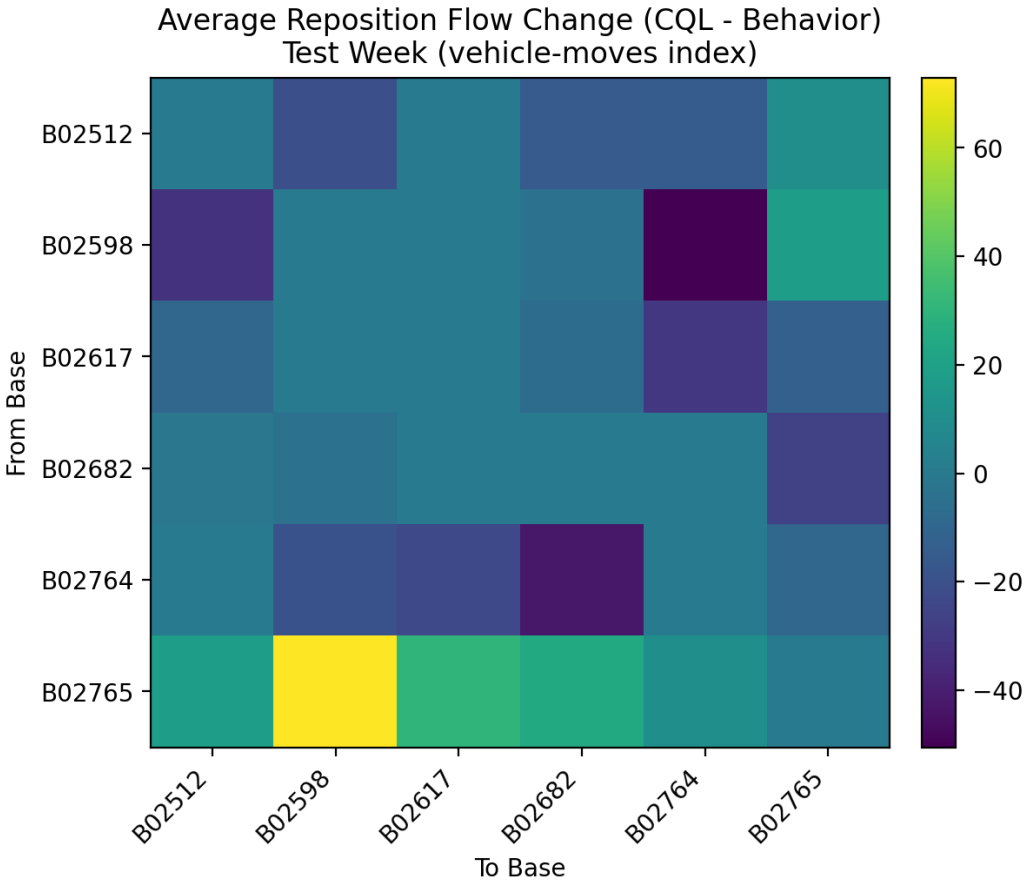

Figure 6. Difference in Implied Reposition Flow Matrix Between CQL and Behavior on the Test Week

Compared with DQN-style fitted Q-learning, CQL produces slightly better waiting-time proxy and lower inequality in unmet demand, which is consistent with offline RL theory: a non-conservative Q learner can overestimate values of rarely observed actions and become overconfident, whereas CQL penalizes such actions and tends to stay closer to the behavior support (Levine et al., 2020; Kumar et al., 2020). In this paper, we therefore interpret CQL primarily as a conservative stabilizer rather than as evidence that offline RL discovers a broadly superior dispatching strategy. We would expect BC and CQL to diverge more clearly with more training days, more diverse behavior actions, a larger action set, or more volatile demand shares.x

G. Flow Interpretation

Journal of Technology Informatics and Engineering (JTIE)
Vol. 4 No. 2 August 2025
E-ISSN: 2961-9068; P-ISSN: 2961-8215, Pages 337-363
Journal Link: https://jtie.stekom.ac.id/index.php/jtie/index

Figure 6 visualizes the difference between CQL and behavior in an implied flow matrix derived from day-to-day distribution changes. Because the dataset is aggregated, this matrix should be interpreted as an index of the amount of mass that must be redistributed between bases under each policy, rather than as literal vehicle routes. CQL reduces several cross-base transfers present in the historical baseline and concentrates adjustments on fewer moves, explaining its lower deadhead index. From an operational perspective, such a policy could involve offering fewer but more targeted incentives for drivers to reposition, or selecting stable "modes" of allocation that avoid oscillations.

H. Validation-Week Consistency

We also evaluated policies during the validation week (Feb 15–21, 2015) to select forecasting checkpoints and verify controller stability before testing. Qualitatively, policy rankings were similar: Uniform remained worst, NoReposition and behavior performed strongly, and CQL reduced deadhead relative to PredDemandHeuristic. This consistency is important because it indicates that the observed test-week behavior is not a single-week artifact, despite the limited horizon.

I. Ablation: Forecasting-Control Coupling

Table 10 compares heuristic and CQL controllers under different demand predictors. Heuristic demand-proportional control is highly sensitive: when predictions are noisy or biased, the controller responds by moving large supply mass, increasing deadhead and inequality. In contrast, CQL shows similar performance under either predictor because it selects from a conservative action set and value function learned from historical behavior. This supports a practical recommendation: when integrating forecasting with control, it is often safer to use conservative controllers that dampen response to prediction noise.

Table 10. Ablation: Impact of Demand Predictor and Controller on Test-Week Performance

DemandPredictor	Controller	AcceptanceRate	AvgWait(min)	DeadheadMiles	GiniUnmet
ST-Transformer	Heuristic	0.9190	4.277	1499.691	0.5110
TCN	Heuristic	0.9146	4.637	1864.962	0.6032
ST-Transformer	CQL	0.9146	4.329	577.076	0.5112
TCN	CQL	0.9146	4.329	577.076	0.5112

J. Sensitivity Analysis

Table 11 confirms that the deadhead-mile index scales linearly with the assumed miles-per-move coefficient, as expected by construction. Table 12 further shows that both acceptance rate and the waiting-time proxy depend on the capacity surrogate c: higher assumed per-vehicle capacity reduces unmet demand and improves acceptance. During the training period, the utilization ratio D/Vexhibits a median of 8.38 with an interquartile range of 7.19–9.22, while base-specific

medians range from 6.79 (B02512) to 8.82 (B02682). Accordingly, $c = 8.38$is interpreted as a pooled surrogate rather than a base-invariant constant, and the results in Table 12 should be viewed as calibration sensitivity rather than parameter estimation.

For the waiting-time proxy, $w_{\min}$functions as an additive offset that affects only the absolute reported values without altering policy ranking, whereas w_{slope}rescales the overload penalty when $\rho > 1$. Consequently, $w_{\min}$primarily influences level calibration, while $c$and w_{slope}determine the severity of penalties during overload conditions.

Because the scalar reward aggregates acceptance, waiting time, deadhead, and inequality (Gini), we also evaluate nearby alternative weight configurations. Under a service-oriented setting (0.03 for waiting time, 0.0003 for deadhead, and 0.05 for Gini) and a cost-oriented setting (0.015, 0.0008, 0.15), the Uniform policy consistently remains the worst performer, and aggressive demand-proportional reallocation remains suboptimal once deadhead costs are meaningfully penalized. These results indicate that precise scalar ranking is less critical than the broader Pareto trade-off between service quality and repositioning cost.

Finally, inequality metrics may become unstable when unmet demand approaches zero. In such cases, alternative fairness measures, such as the dispersion of waiting-time proxies, would provide more robust evaluation criteria in future work.

Table 11. Sensitivity: Deadhead-Mile Index Under Different Miles-per-Move Coefficients (Test Week)

MilesPerMove	Policy	acceptance	avg_wait	deadhead_miles	gini_unmet
1.000	Behavior	0.9184	4.119	426.731	0.3987
1.000	CQL	0.9146	4.329	288.538	0.5112
2.000	Behavior	0.9184	4.119	853.461	0.3987
2.000	CQL	0.9146	4.329	577.076	0.5112
3.000	Behavior	0.9184	4.119	1280.192	0.3987
3.000	CQL	0.9146	4.329	865.614	0.5112

Table 12. Sensitivity: Service Outcomes Under Different per-Vehicle Capacity Surrogates (Test Week, Miles-per-Move=2)

CapacityTripsPerVehicle	Policy	acceptance	avg_wait	deadhead_miles	gini_unmet
7.546	Behavior	0.8305	5.522	853.461	0.3739
7.546	CQL	0.8294	5.723	577.076	0.3572
8.384	Behavior	0.9184	4.119	853.461	0.3987
8.384	CQL	0.9146	4.329	577.076	0.5112
9.222	Behavior	0.9781	3.286	853.461	0.3989
9.222	CQL	0.9614	3.613	577.076	0.7465

V. DISCUSSION

A. Limitations

Our study is intentionally constrained to base-level daily aggregates. This limitation affects both prediction and control. First, the forecast horizon is one day, which is coarser than typical operational decision cycles. Second, the digital twin does not model travel times, congestion, or within-day dynamics, and it treats supply totals as exogenous. Third, the deadhead-mile index is a proxy and likely overestimates true empty mileage. Despite these limitations, the pipeline is useful as a reproducible benchmark for end-to-end integration and for understanding how forecast uncertainty propagates into control decisions under conservative vs aggressive policies.

B. Implications for Practice

The strongest qualitative lesson is that, at daily base-level aggregates, gains arise primarily from avoiding overreaction to fairly stable demand shares rather than from complex dynamic control. Controlling the sensitivity of repositioning to forecast changes is therefore at least as important as improving forecast accuracy. Demand-proportional heuristics effectively implement a high-gain controller: small prediction noise becomes large spatial reallocation. Offline RL with an action-constrained policy class behaves more like a low-gain controller: it selects among stable historical modes and therefore avoids oscillations. This suggests that, in production, conservative control can provide robustness even when prediction models are imperfect.

VI. CONCLUSION AND RECOMMENDATION

We developed an end-to-end digital-twin dispatching pipeline using the public FOIL dataset Uber-Jan-Feb-FOIL.csv. The pipeline integrates spatio-temporal demand forecasting with offline RL for next-day fleet repositioning and evaluates policies using acceptance rate, waiting-time proxy, deadhead-mile index, and imbalance metrics. On this FOIL-scale benchmark, a compact TCN achieved the best test accuracy, although the aggregate advantage is driven primarily by the dominant base B02764 and should not be over-generalized to all bases. Within the control module, conservative offline RL (CQL) produced stable closed-loop performance and lower repositioning aggressiveness than naive demand-proportional heuristics, but in this very small offline dataset it coincided with BC. We therefore interpret the RL result as evidence that conservative, behavior-close control can be useful in coarse aggregate settings, not as evidence of general offline-RL superiority for urban dispatching.

Based on these results, we recommend three practices for mobility teams implementing prediction-to-control pipelines. First, validate demand predictors with strict chronological splits and report per-base errors, because average metrics can hide minority-base failures that drive inequitable outcomes. Second, avoid brittle control laws that translate prediction changes directly into large rebalancing actions; instead, use conservative offline RL, action constraints, or explicit smoothing to limit oscillations and deadhead. Third, document digital twin assumptions and

conduct sensitivity analyses, especially when metrics depend on surrogate parameters such as per-vehicle capacity, waiting-time-proxy hyperparameters, miles-per-move, or reward weights. Transparent assumptions improve trust and make it easier to compare results across studies.

The most important extension is higher-resolution and longer-horizon data. Hourly demand and supply, spatial grids, and travel-time models would enable a more realistic simulator with explicit queuing and routing, and would allow repositioning at operational time scales. Longer datasets with more diverse behavior actions would also provide a more meaningful test bed for separating BC from conservative offline RL methods such as CQL. Future studies can broaden offline RL comparisons to include BCQ (Fujimoto et al., 2019) and IQL (Kostrikov et al., 2021), and can incorporate fairness-aware objectives that balance service across neighborhoods. Finally, integrating incentive costs and driver response models would connect repositioning policies to real-world implementability.

Dispatch and repositioning decisions can unintentionally concentrate service in already well-served areas while degrading access elsewhere. Even in our simplified twin, imbalance metrics vary across policies, illustrating that efficiency-focused objectives can have distributional consequences. When extending this work to finer spatial granularity, teams should consider explicit fairness constraints or multi-objective optimization that caps disparities in wait proxies or acceptance rates across neighborhoods.

REFERENCES

Agustin, N., & Rahayu, S. (2025). The Role of Social Media as a Micro-Ecosystem in Supporting Community-Based E-Learning Platforms: A Systematic Literature Review. *Journal of Technology Informatics and Engineering, 4*(3), 369–390. https://doi.org/10.51903/jtie.v4i3.445

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-Demand High-Capacity Ride-Sharing via Dynamic Trip-Vehicle Assignment. *Proceedings of the National Academy of Sciences*, *114*(3), 462–467. https://doi.org/10.1073/pnas.1611675114

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv Preprint*, arXiv:1803.01271. https://arxiv.org/abs/1803.01271

Batty, M. (2018). Digital Twins. *Environment and Planning B: Urban Analytics and City Science*, *45*(5), 817–820. https://doi.org/10.1177/2399808318796416

Boschert, S., & Rosen, R. (2016). Digital Twin—The Simulation Aspect. In *P. Hehenberger & D. Bradley (Eds.),* Mechatronic Futures, 59–74. https://doi.org/10.1007/978-3-319-32156-1_5

FiveThirtyEight. (2015). *Uber TLC FOIL Response (NYC): Uber-Jan-Feb-FOIL.csv*. GitHub. https://github.com/fivethirtyeight/uber-tlc-foil-response

Fujimoto, S., Meger, D., & Precup, D. (2019). Off-Policy Deep Reinforcement Learning Without Exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, *97*, 2052–2062. https://proceedings.mlr.press/v97/fujimoto19a.html

Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, *8*, 108952–108971. https://doi.org/10.1109/access.2020.2998358

Gini, C. (1921). Measurement of Inequality of Incomes. *The Economic Journal*, *31*(121), 124–126. https://doi.org/10.2307/2223319

Grieves, M. (2014). *Digital Twin: Manufacturing Excellence Through Virtual Factory Replication* (White Paper). Apriso / Dassault Systèmes. https://www.manufacturingview.media/uploads/resources/resources/222025114804PM_Resource_DigitalTwinManufacturingExcellenceThroughVirtualFactoryReplication.pdf

Holler, J., Vuorio, R., Qin, Z., Tang, X., Jiao, Y., Jin, T., Singh, S., Wang, C., & Ye, J. (2019). Deep Reinforcement Learning for Multi-Driver Vehicle Dispatching and Repositioning Problem. *arXiv Preprint*, arXiv:1911.11260. https://arxiv.org/abs/1911.11260

Jiao, Y., Tang, X., Qin, Z. T., Li, S., Zhang, F., Zhu, H., & Ye, J. (2021). Real-World Ride-Hailing Vehicle Repositioning Using Deep Reinforcement Learning. *Transportation Research Part C: Emerging Technologies*, *130*, 103289. https://doi.org/10.1016/j.trc.2021.103289

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980

Kušić, K., Schumann, R., & Ivanjko, E. (2023). A Digital Twin in Transportation: Real-Time Synergy of Traffic Data Streams and Simulation for Virtualizing Motorway Dynamics. *Advanced Engineering Informatics*, *55*, 101858. https://doi.org/10.1016/j.aei.2022.101858

Ke, J., Zheng, H., Yang, H., & Chen, X. M. (2017). Short-Term Forecasting of Passenger Demand under On-Demand Ride Services: A Spatio-Temporal Deep Learning Approach. *Transportation Research Part C: Emerging Technologies*, *85*, 591–608. https://doi.org/10.1016/j.trc.2017.10.016

Kleinrock, L. (1975). Queueing Systems. Volume 1: Theory (Leonard Kleinrock). *SIAM Review*, *18*(3), 512-514. https://doi.org/10.1137/1018095

Kostrikov, I., Nair, A., & Levine, S. (2021). Offline Reinforcement Learning with Implicit Q-Learning. *arXiv Preprint*, arXiv:2110.06169. https://arxiv.org/abs/2110.06169

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in Manufacturing: A Categorical Literature Review and Classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative Q-Learning for Offline Reinforcement Learning. *Advances in Neural Information Processing Systems*, *33*, 1179-1191. https://proceedings.neurips.cc/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf

Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020). Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv Preprint*, arXiv:2005.01643. https://arxiv.org/abs/2005.01643

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations (ICLR 2018)*. https://openreview.net/forum?id=H1g-f2CcKX

Lin, K., Zhao, R., Xu, Z., & Zhou, J. (2018). Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* 1774–1783. https://doi.org/10.1145/3219819.3219993

Lim, B., Arik, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting. *International Journal of Forecasting*, *37*(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2020.05.011

MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations**.** In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1*, 281–297. https://projecteuclid.org/euclid.bsmsp/1200512992

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-Level Control through Deep Reinforcement Learning. *Nature*, *518*, 529–533. https://doi.org/10.1038/nature14236

Musrifah, F., & Hasanah, I. A. (2025). Ethical Implications of AI-Driven Recruitment: A Multi-Perspective Study on Bias and Transparency in Digital Hiring Platforms. *Journal of Management and Informatics, 4*(1), 599–616. https://doi.org/10.51903/jmi.v4i1.140

Qin, Z. T., Zhu, H., & Ye, J. (2022). Reinforcement Learning for Ridesharing: An Extended Survey. *Transportation Research Part C: Emerging Technologies*, *144*, 103852. https://doi.org/10.1016/j.trc.2022.103852

Rokhman, N., Sumaryanto, S., & Maulan, P. A. (2025). Perancangan Tempat Sampah Cerdas Berbasis Arduino UNO di MTS Sunan Kalijaga. *JUISI: Jurnal Ilmiah Sistem Informasi, 4*(1), 1–12. https://doi.org/10.51903/twpgeq37

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press. http://incompleteideas.net/book/the-book-2nd.html

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital Twin-Driven Product Design, Manufacturing and Service with Big Data. *The International Journal of Advanced Manufacturing Technology*, *94*, 3563–3576. https://doi.org/10.1007/s00170-017-0233-1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems*, *30*, 1-11. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., & Xiong, H. (2020). Spatial-Temporal Transformer Networks for Traffic Flow Forecasting. *arXiv*. https://arxiv.org/abs/2001.02908

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. https://www.ijcai.org/proceedings/2018/

Zhou, M., Jin, J., Zhang, W., Qin, Z., Jiao, Y., Wang, C., Wu, G., Yu, Y., & Ye, J. (2019). Multi-Agent Reinforcement Learning for Order-Dispatching via Order-Vehicle Distribution Matching. *arXiv*. https://arxiv.org/abs/1910.02591