International Multidisciplinary Journal of Emerging Technologies and Applications (IMJETA)
Vol. 1, No. 4, pp. 50-67, July 31, 2026
Received July 09, 2026; Revised July 18, 2026; Accepted July 27, 2026
Published July 31, 2026

Domain Adaptation for CI/CD Workload Prediction: Transferring Seasonality Patterns Across Non-Overlapping Time Windows

Juan Camilo Escobar Naranjo [0009-0008-3262-1668] and Nestor Darío Duque-Méndez [0000-0002-4608-281X]

Department of Informatics and Computing, Universidad Nacional de Colombia, Manizales, Colombia

jcescobarn@unal.edu.co, ndduqueme@unal.edu.co

Abstract. Machine learning models for workload prediction typically require training data from the same time period as deployment. In CI/CD environments, obtaining representative real-world training data is often difficult due to privacy concerns, data availability constraints, or the need to predict for newly established infrastructure. This paper presents a transfer learning methodology that enables transferring seasonality patterns and workload characteristics from real billing data collected during one-time period to real workflow execution logs from a completely non-overlapping period. The article approach combines two techniques: (1) Quantile Mapping to align duration distributions between the target- and source-domain data, preserving structural characteristics (shape, variability, skewness), and (2) Cross-Temporal Seasonality Transfer to map weekly activity patterns across time windows. Validated on GitHub Actions data spanning two distinct quarters (January-April 2023 billing data transferred to June-October 2023 real execution logs), our method achieved a strong correlation ($r = 0.8739$, $p = 0.0101$) for weekly patterns, explaining 76.4% of the variance in seasonality ($R^2 = 0.764$), and aligned the duration distribution, reducing the CV error from 0.26 to 0.01, skewness from 1.06 to 0.03, and kurtosis from 2.33 to 0.07 relative to the source domain – all without overlapping temporal coverage between the two data sources.

Keywords: Transfer Learning; Domain Adaptation; Workload Prediction; Seasonality Patterns; Continuous Integration

1. Introduction

1.1. Motivation

Machine Learning (ML) applications in systems and infrastructure management face a persistent challenge: obtaining sufficient training data that accurately represents production workloads (Weiss et al., 2016; Rossi et al., 2019). In Continuous Integration/Continuous Deployment (CI/CD) environments, this challenge is particularly acute due to:

- Privacy constraints: organizations rarely share internal build logs containing proprietary code structure and development practices
- Cold-start problem: new teams or infrastructure lack historical data for model training
- Data heterogeneity: available datasets (synthetic or academic) may not match production characteristics
- Temporal gaps: billing records may exist from one period while detailed logs are only available for another

Traditional ML approaches assume that training and deployment data are drawn from the same distribution – an assumption frequently violated in practice. Transfer learning offers a solution by enabling knowledge transfer from a source domain to a related but different target domain (Pan & Yang, 2010).

1.2. The CI/CD Data Challenge

Consider a realistic scenario: an organization has GitHub Actions billing data from Q1 2023 showing usage patterns, but detailed workflow execution logs (with timing, concurrency, job sequences) only exist for Q3 2023. Directly training an auto scaling ML model requires merging these heterogeneous data sources despite their non-overlapping time periods and different granularities.

This paper addresses the question: **can we transfer workload characteristics and seasonality patterns from real billing data in one-time period to execution logs from a completely different period, enabling accurate predictive model training?**

1.3. Paper Organization

Section 2 reviews related work on transfer learning, domain adaptation, and time-series seasonality. Section 3 formalizes the problem and presents our transfer learning methodology. Section 4 details the experimental setup and datasets. Section 5 presents quantitative validation results. Section 6 discusses limitations and future directions. Section 7 concludes.

1.4. Contribution to the Sustainable Development Goals

Beyond its methodological contribution, this work is aligned with the United Nations 2030 Agenda for Sustainable Development (United Nations, 2015). Two

Sustainable Development Goals (SDGs) are directly relevant to the technical domain addressed here, namely machine learning applications for CI/CD infrastructure, cloud computing efficiency, and resource optimization. First, SDG 9 (Industry, Innovation, and Infrastructure), and specifically Target 9.4 on upgrading infrastructure and retrofitting industries for greater resource-use efficiency, is advanced by a methodology that reduces resource misallocation and prediction error in autoscaling cloud infrastructure, allowing organizations to provision computing capacity that more closely matches real demand. Second, SDG 12 (Responsible Consumption and Production), and specifically Target 12.2 on the sustainable management and efficient use of natural resources, is supported because more accurate workload prediction helps prevent the over-provisioning of compute resources, which in turn is associated with lower energy consumption and a smaller carbon footprint for the data centers that host CI/CD infrastructure (Beloglazov et al., 2011; Subirats & Guitart, 2015). We return to these practical and environmental implications in Section 6.4.

2. Background and Related Work

2.1. Transfer Learning Fundamentals

Pan and Yang (2010) established the theoretical foundation for transfer learning, defining it as the ability to apply knowledge gained in one setting (source domain D_S) to a different but related setting (target domain D_T). Success depends on identifying invariant features – characteristics that remain stable across domains.

Transfer learning is categorized into:

- Domain Adaptation: source and target have different distributions but the same task
- Task Transfer: source and target share distributions but differ in prediction tasks
- Feature Transfer: shared representation learning across domains

Our work falls under domain adaptation: both the source (billing data) and target (execution logs) aim to characterize CI/CD workload patterns, but differ in time period, granularity, and data schema.

2.2. Domain Adaptation in Time Series

Ben-David et al. (2010) formalized domain adaptation theory, showing that transfer success depends on minimizing the divergence between the source and target domains.

For time series, Hyndman and Athanasopoulos (2021) distinguish between:

- Trend: long-term direction
- Seasonality: periodic patterns (daily, weekly, yearly)
- Cycle: non-periodic oscillations

- Residual: irregular variation

2.3 Quantile Mapping for Distribution Alignment

Quantile mapping, widely used in climate science for bias correction (Cannon et al., 2015), transforms data from one distribution to match another while preserving rank order. Given a source distribution F_S and a target distribution F_T, quantile mapping is defined as show in Ec. 1.

$$x_{mapped} = F_T^{-1}(F_S(x)) \qquad (1)$$

This approach preserves percentile structure: the median of the source data maps to the median of the target data, the 90th percentile to the 90th percentile, and so on.

Patki et al. (2016) applied similar principles in the Synthetic Data Vault, demonstrating that distribution-preserving transformations enable realistic synthetic data generation. We extend this idea to cross-temporal transfer in CI/CD contexts.

2.4. Seasonality in Software Data Series

Limited prior work analyzes temporal patterns in CI/CD systems. Moriconi et al. (2024) documented weekly cycles in GitHub Actions usage, with peak activity Tuesday-Thursday and minimal weekend activity. However, they did not investigate whether these patterns are invariant across time periods.

Claes et al. (2018) analyzed commit timestamps across 86 open-source projects, finding that two-thirds of developers predominantly follow standard office hours (10h-18h) and reduce activity at night and on weekends. Their work suggests that seasonality reflects human behavior rather than technical factors.

2.5. Correlation as Transfer Learning Validation

Cohen (1988) established effect-size thresholds for correlation in the behavioral sciences: small ($r = 0.10$), medium ($r = 0.30$), and large ($r = 0.50$). Applied time-series and model-validation literature has derived additional practical thresholds:

- $r > 0.7$: strong correlation (high pattern similarity)
- $0.5 < r \leq 0.7$: moderate correlation
- $r \leq 0.5$: weak correlation (insufficient for transfer)

Weiss et al. (2016) argue that a correlation $r > 0.7$ between source- and target-domain features indicates sufficient similarity for successful knowledge transfer – a threshold consistent with Cohen's large effect ($r \geq 0.50$) and widely adopted in domain-adaptation validation. We adopt this criterion to evaluate our seasonality transfer.

2.6. Transfer Learning in Time Series and Forecasting

Beyond the foundational survey of Pan and Yang (2010), Zhuang et al. (2021) provide a broader synthesis of transfer learning mechanisms, categorizing approaches by what is transferred (instances, features, parameters, or relations) rather than only by domain relationship. Within forecasting specifically, Jin et al. (2022) propose an attention-sharing architecture that transfers statistical strength from a data-rich source series to a data-scarce target series, addressing the same core scarcity problem that motivates our billing-to-execution-log transfer, though through representation learning rather than distributional and seasonal alignment. Classical decomposition methods remain a natural complement to these transfer approaches: Cleveland et al. (1990) introduced the STL procedure for separating trend, seasonal, and residual components in time series, and Box et al. (2015) formalized the ARIMA family that underlies much of the workload-forecasting literature cited below.

For sequence models capable of capturing longer-range temporal dependence, Hochreiter and Schmidhuber (1997) introduced the long short-term memory (LSTM) architecture, widely used as a baseline in subsequent CI/CD and cloud-workload prediction work. Non-parametric ensemble methods, most notably Random Forests (Breiman, 2001), are also common baselines in this literature and appear in several of the workload-prediction studies reviewed in Section 2.7.

2.7. Machine Learning for Cloud Workload Prediction and Autoscaling

Predictive autoscaling is the broader application context for this work. Calheiros et al. (2014) were among the first to apply ARIMA models to cloud workload prediction, showing that forecast-informed provisioning improves quality-of-service compared to reactive, threshold-based scaling. Saxena et al. (2023) survey machine-learning-centered workload prediction models for the cloud, organizing the field by statistical, machine-learning, and deep-learning approaches, and note that most published models are trained and evaluated within a single, contiguous data-collection window — the same limitation our cross-temporal transfer methodology is designed to relax.

Rossi et al. (2019), already cited in Section 2.1, similarly observe that autoscaling techniques generally assume representative historical data is available for the deployment environment. Closer to our specific domain, Bisong et al. (2017) modeled CI build durations from project and commit metadata, Chen et al. (2020) proposed history-aware build outcome prediction for fast feedback in continuous integration, and Saidani et al. (2022) applied deep learning to the related problem of predicting CI build failures; together, these studies confirm that CI/CD temporal data exhibits learnable structure, but none of them addresses the specific problem of transferring that structure across non-overlapping time windows. Vasilescu et al. (2015) provide complementary evidence, at the level of development practice rather

than infrastructure, that continuous integration adoption measurably changes - yet does not eliminate — the weekly rhythm of developer activity that our seasonality — transfer stage exploits.

2.8. Quantile Mapping and Distributional Bias Correction

Section 2.3 introduced quantile mapping through Cannon et al. (2015). The technique has a longer history in climate bias correction that is directly relevant to how we justify its use here: Maraun (2013) shows that naive quantile mapping can distort the temporal and spatial structure of a corrected series when source and target resolutions differ, a caution we address in Section 6.3 by restricting the transferred features to distributional and weekly-aggregate statistics rather than fine-grained sequences. This body of work, developed for an entirely different application domain (climate science), further supports the domain-adaptation argument of Section 2.1: an invariant-preserving statistical transformation can be reused across domains that share little else in common.

2.9. Energy Efficiency and Sustainable Resource Management in Cloud Infrastructure

Finally, the sustainability motivation introduced in Section 1.4 has a substantial supporting literature. Beloglazov et al. (2011) provide an early taxonomy of energy-efficient data center design spanning hardware, virtualization, and scheduling layers, and argue that accurate demand estimation is a prerequisite for energy-proportional operation. Subirats and Guitart (2015) develop methods to assess and forecast energy efficiency specifically for cloud computing platforms, reinforcing the link — assumed in Section 1.4 — between workload-prediction accuracy and reduced over-provisioning. Mao et al. (2016) show that reinforcement-learning-based resource management can further reduce waste relative to heuristic scheduling once accurate demand signals are available, illustrating a complementary direction to the transfer learning approach developed in this paper. Dean and Barroso (2013) further note that over-provisioning is often a rational response to tail latency rather than pure inefficiency, a nuance that qualifies the resource-savings argument of Section 1.4 and that we revisit briefly in Section 6.3.

3. Methodology

3.1. Problem Formalization

Datasets. We work with two datasets.

Source Domain D_S (Real Billing Data):

- Time period: January 12 – April 12, 2023 (91 days)
- Records: 63,324 billing line items
- Granularity: daily aggregation by repository and SKU
- Key fields: date, repository, SKU, minutes used, cost
- Characteristics: real organizational usage patterns

Target Domain D_T (Real Execution Logs):

- Time period: June 30 – October 1, 2023 (93 days)
- Records: 100,000 workflow executions
- Granularity: individual job-level timing (millisecond precision)
- Key fields: repository, workflow name, start time, duration, runner type
- Characteristics: real GitHub Actions logs collected from public repositories (GHALogs; Moriconi et al., 2024)

Challenges:

1. Non-overlapping time: no temporal intersection between datasets
2. Different schemas: billing data lacks job-level detail; logs lack cost information
3. Different distributions: target durations may not match source patterns
4. Heterogeneous granularity: daily aggregates versus millisecond timestamps
5. Transfer Learning Objective. Our goal is to enrich D_T with features from D_S so that a predictive model trained on the enriched dataset performs equivalently to one trained on complete real execution logs (unavailable).

3.2. Transfer Learning Pipeline

The article methodology consists of four sequential stages (Table 1).

Table 1. Cross-temporal transfer learning pipeline for CI/CD.

Cross-Temporal Transfer Learning Pipeline
D_S (Q1 Billing Data) → **Stage 1: Preprocessing**
↓
Stage 2: Quantile Mapping (distributional alignment)
↓
Stage 3: Seasonality Extraction (weekly patterns)
↓
Stage 4: Feature Transfer → D'_T (enriched Q3)

Real billing data (source domain D_S) enriches the target-domain execution logs (D_T) through four sequential stages, producing a dataset D'_T with aligned distributional and seasonal characteristics.

Stage 1: Data Preprocessing.

Billing Data Preprocessing. GitHub Actions data is exported in CSV format with UTF-8 BOM encoding (Pintye et al., 2024; Kinsman et al., 2022). Records for the "Actions" product are filtered and usage minutes are aggregated by (date, repository). The temporal features day_of_week and is_weekend is extracted from the date column, following standard trend/seasonality/residual decomposition conventions (Hyndman & Athanasopoulos, 2021). Mean daily concurrency is estimated as $c^- = M_{total}/(24 \times 60)$, where M_{total} is the total minutes billed for the day,

representing the average number of simultaneously active runners (Pintye et al., 2024).

Source Domain Preprocessing steps:

1. Load CSV with UTF-8 BOM encoding (Pintye et al., 2024)
2. Filter for product = "Actions" (Kinsman et al., 2022)
3. Aggregate minutes by (date, repository)
4. Extract day_of_week and is_weekend from the date column (Hyndman & Athanasopoulos, 2021)
5. Compute $c^- = M_{total}/(24 \times 60)$ (mean daily concurrency)

Execution Log Preprocessing. GHALogs (Moriconi et al., 2024) are distributed in JSON Lines format, where each record represents an individual job execution with started_at and completed_at fields at millisecond precision. Only jobs with status = ''completed'' are retained, excluding cancelled or failed runs to avoid distorting the duration distribution. Instantaneous concurrency is obtained by resampling start/end events into 1-minute intervals and counting active jobs within each window, a standard technique for irregular time-series analysis (Hyndman & Athanasopoulos, 2021).

Targe Domain Preprocessing steps:

1. Parse the JSON Lines format (Moriconi et al., 2024)
2. Extract started_at and completed_at for each job
3. Compute duration = $completed\_at - started\_at$
4. Filter status = ''completed'' (exclude cancelled/failed)
5. Resample to 1-minute intervals, counting active jobs per window (Hyndman & Athanasopoulos, 2021)

Stage 2: Quantile Mapping for Duration Alignment.
Target-domain durations may not reflect real-world variability. We apply quantile mapping to adjust D_T durations using D_S characteristics.

Algorithm 1: Quantile Mapping.

- Require**:** source durations D_S, target durations D_T.
- Ensure**:** corrected target durations D'_T.
    - Compute quantiles $Q_S = \{q_{0.01}, q_{0.02}, \ldots, q_{0.99}\}$ of D_S
    - Compute quantiles Q_T of D_T
    - For each duration $d_t \in D_T$: find its percentile $p = F_{D_T}(d_t)$ and map it to the source quantile $d'_t = F^{-1}_{D_S}(p)$
    - Return D'_T

This transformation preserves:

- Rank order: the fastest jobs remain fastest, the slowest remain slowest
- Distributional shape: skewness and kurtosis match the real data

- Extreme values: 99th-percentile durations reflect real outliers

Stage 3: Seasonality Pattern Extraction.
We extract weekly activity patterns from both datasets.

For Billing Data: we compute the average concurrency pattern per day of week, aggregating across all weeks and normalizing to the maximum observed value.

For Execution Logs: we apply the same per-day-of-week aggregation and normalization procedure.

Correlation Validation: we compute the Pearson correlation between the normalized patterns of both sources. If $r > 0.7$, we proceed with transfer; otherwise, the domains are considered too dissimilar.

Stage 4: Feature Transfer.
We inject features from D_S into D_T.

Transferred Features:
- day_of_week: mapped from timestamp (0 = Monday, 6 = Sunday)
- hour: extracted from timestamp (0-23)
- is_weekend: binary indicator (Saturday/Sunday)
- period_of_day: categorical (morning/afternoon/evening/night)
- lookback_1min: average concurrency in the preceding 60 seconds

These features encode the organizational seasonality validated in Stage 3.

3.3. Validation Methodology

- Distributional Metrics. We assess distributional similarity using the coefficient of variation (CV, σ/μ; target: $CV_T \approx CV_S$), skewness (positive skew indicates a long right tail, i.e., few very long jobs), and kurtosis (high kurtosis indicates frequent extreme values).
- Temporal Pattern Validation. We report the seasonality correlation (Pearson r between weekly patterns), R^2 for explained variance (quantifying how much of the target seasonality is explained by the source seasonality), and statistical significance (two-tailed t-test for correlation; target: $p < 0.05$).

4. Experimental Setup

4.1. Data Processing Pipeline

- Hardware. CPU: Intel Xeon W-10855M @ 2.80GHz; RAM: 16 GB; storage: SSD; OS: Windows 11 Pro.

- Software. Python 3.9; pandas 1.5.3 (data manipulation); numpy 1.24.2 (numerical operations); scipy 1.10.1 (statistical tests); scikit-learn 1.2.2 (quantile mapping).

4.2. Dataset Statistics

Table 2 presents key statistics for both datasets.

Table 2. Dataset Characteristics

Characteristic	Billing (Q1)	Logs (Q3)
Time period	Jan-Apr 2023	Jun-Oct 2023
Records (CSV lines)	63,324	100,000
Unique days	91	93
Workflow executions*	63,601	100,000
Granularity	Daily aggregate	Job-level
Duration statistics (reconstructed for Q1):		
Mean duration	4.32 min	3.87 min
Median duration	2.15 min	1.92 min
Standard deviation	6.21 min	4.89 min
95th percentile	15.8 min	12.3 min
Maximum duration	89.2 min	67.4 min

*The 63,324 CSV lines correspond to (date, repository, SKU) aggregations; the 63,601 workflows are inferred by counting unique executions after disaggregating multi-repository records. Q1 durations are approximated from billing minutes divided by the estimated job count; this limitation is discussed in Section 6.3.

4.3. Quantile Mapping Configuration

We use 99 quantiles (1st, 2nd, ..., 99th percentile) for fine-grained mapping. For values outside the observed range, we apply linear extrapolation from the two nearest quantiles.

5. Results

5.1. Quantile Mapping Validation

Table 3 shows distributional metrics before and after quantile mapping.

Table 3. Distributional Alignment Results

Metric	Source (Q1)	Target Before	Target After
Mean (min)	4.32	3.87	4.28
Median (min)	2.15	1.92	2.13
Std. dev. (min)	6.21	4.89	6.18
CV	0.68	0.42	0.67
Skewness	1.24	0.18	1.21
Kurtosis	3.45	1.12	3.38
95th percentile	15.8	12.3	15.6

Key Findings

1. Coefficient of Variation. After mapping, CV increased from 0.42 to 0.67, closely matching the source CV of 0.68. This indicates that target-domain data (post-mapping GHALogs) now exhibits realistic variability.
2. Skewness. Skewness increased from 0.18 to 1.21 (versus source 1.24), demonstrating that long-duration outliers are now properly represented. The near-zero original skewness indicates that the original GHALogs distribution was excessively uniform relative to real workload.
3. Kurtosis. Kurtosis increased from 1.12 to 3.38 (versus source 3.45), indicating heavier tails matching real data. This captures the presence of occasional very long-running jobs.
4. Percentile Alignment. The 95th percentile shifted from 12.3 to 15.6 minutes, nearly matching the 15.8-minute source value. This ensures that extreme values are realistically scaled.

5.2. Seasonality Transfer Validation

Table 4 shows normalized weekly patterns for both datasets. For Q1 2023 billing data (source domain) and Q3 2023 GHALogs (target domain). Values are normalized to the weekly maximum. Tue–Thu are peak days; Sat–Sun are the lowest days. Pearson correlation: $r = 0.8739$, $p = 0.0101$.

Table 4. Patterns for both datasets

	Mon	Tue–Thu	Sat–Sun
Billing Q1	0.72	1.00	0.31
GHALogs Q3	0.68	1.00	0.33

Table 5 summarizes the seasonality correlation analysis.

Table 5. Seasonality Correlation Analysis

Metric	Value
Pearson correlation (r)	0.8739
P-value (two-tailed)	0.0101
R^2 (explained variance)	0.764
Cohen's interpretation	Strong
Significance level	$p < 0.05$
Weekly pattern consistency	
Peak days	Tue-Thu (both)
Low days	Sat-Sun (both)
Monday ramp-up	Yes (both)
Friday drop-off	Yes (both)

Critical Result. A Pearson correlation of $r = 0.8739$ ($p = 0.0101$, $n = 7$ days) exceeds the $r > 0.7$ threshold adopted in domain-adaptation validation (Weiss et al., 2016), validating that weekly patterns are invariant across quarters. Given the small sample size ($n = 7$), we performed a complementary hourly-granularity

analysis ($n = 168$ weekly hourly slots), obtaining $r = 0.742$ ($p < 0.001$), confirming the robustness of the finding with greater statistical power.

Qualitative Pattern Analysis. Both datasets exhibit:

- Tuesday-Thursday peaks: maximum activity mid-week
- Weekend lows: 60-70% reduction on Saturday/Sunday
- Monday ramp-up: gradual increase from the weekend baseline
- Friday drop-off: decline as the week ends

This consistency reflects stable organizational work schedules (developers work weekdays, rest on weekends), independent of which quarter is observed.

5.3. Ablation Study

To isolate the contribution of each pipeline stage, we evaluated the fidelity of the transferred patterns under three configurations, using exclusively distributional and seasonal-correlation metrics (Table 6).

Table 6. Ablation of Transfer Stages (Pattern-Fidelity Metrics)

Configuration	Target CV	Skewness	Kurtosis	Seasonality r
Source reference (Q1)	0.68	1.24	3.45	—
A: No transfer (raw GHALogs)	0.42	0.18	1.12	N/A
B: Quantile mapping only (duration alignment)	0.67	1.21	3.38	N/A
C: Full transfer (quantile + seasonality)	0.67	1.21	3.38	0.874

Interpretation

- Quantile Mapping stage (A → B): eliminates the distributional divergence. CV moves from 0.42 to 0.67 (residual error of 0.01 relative to the source), skewness from 0.18 to 1.21, and kurtosis from 1.12 to 3.38, replicating the real long-tail shape.
- Seasonality stage (B → C): validates and incorporates temporal invariance. The correlation $r = 0.874$ between source and target weekly patterns confirms that the organizational cycle is transferable. Without this stage, weekly patterns remain unaligned even though the duration distribution has been corrected.
- Ablation conclusion: both stages are necessary – quantile mapping resolves distributional divergence, while seasonality transfer resolves temporal divergence.

6. Discussion

6.1. Why Seasonality Transfer Works

Our results demonstrate that organizational behavior in software development is temporally invariant. Three factors explain this:

1. Human Work Schedules. Developers work Monday-Friday, regardless of which month or quarter. This creates stable weekly cycles driven by labor practices rather than technical factors.
2. Sprint Synchronization. Most organizations use 1-2 week sprints starting Monday and ending Friday, creating synchronized commit and build patterns.
3. Cultural Norms. Reduced activity on Friday afternoon (preparing for the weekend) and Sunday evening (preparing for Monday) reflects universal professional culture.

These factors remain stable across quarters, enabling cross-temporal transfer. They may not, however, generalize to organizations with:

- Globally distributed teams spanning multiple time zones
- Flexible/asynchronous work schedules
- Climate-dependent industries (e.g., agriculture, tourism)

6.2. Quantile Mapping versus Mean/Standard-Deviation Normalization

Traditional normalization scales data using $(x - \mu)/\sigma$, which preserves relative distances but assumes Gaussian distributions. CI/CD durations are highly skewed (long-tailed), making quantile mapping more suitable:

Advantages:

- Distribution-agnostic (works for any shape)
- Preserves rank order
- Correctly handles extreme outliers
- Matches all percentiles simultaneously

Disadvantage: requires sufficient data to estimate quantiles accurately (typically $n > 1000$).

6.3. Limitations and Threats to Validity

- Internal Validity. Temporal Autocorrelation: CI/CD workflows are not independent – a commit triggering build 1 may cause build 2 (dependent tests). Our lookback feature partially captures this, but alternative sequence modeling could improve accuracy. Train/Test Contamination: we use a temporal split (train on early data, test on later data) to prevent leakage. Random splits would overestimate performance due to temporal correlation.

- External Validity. Single Platform: we validated only on GitHub Actions. Transfer to GitLab CI, Jenkins, or CircleCI requires verification (future work). Industry Domain: results may differ for 24/7 operations (e.g., financial services with nightly deployments), seasonal businesses (e.g., retail with holiday spikes), and academic institutions (reduced summer activity).
- Construct Validity. Pearson Correlation Assumptions: Pearson r assumes linear relationships and no extreme outliers. We validated with Spearman rank correlation ($\rho = 0.863$) as a robustness check – results were consistent. Statistical Significance: with $n = 7$ days, the 95% confidence interval for $r = 0.8739$ is approximately $[0.45, 0.97]$, illustrating the width of the interval with small samples. The complementary hourly-granularity analysis ($n = 168$, $r = 0.742$, $p < 0.001$), reported in Section 5.2, provides statistically stronger evidence.

6.4. Practical Deployment Considerations

- Cold-Start Acceleration. Organizations with new or insufficiently historical CI/CD infrastructure can immediately obtain a training dataset with realistic distributions and aligned seasonal patterns, using only billing data from the available period and public GHALogs. This eliminates the wait for representative data accumulation, which typically requires several quarters.
- Privacy Preservation. Transfer learning enables the use of external public data without exposing proprietary internal logs. Only seasonality patterns (weekly cycles) need alignment, not sensitive workflow details.
- Continuous Validation. Deploy with monitoring to detect distribution drift (e.g., organizational restructuring changing work patterns). Monthly correlation checks between current and historical patterns can trigger retraining.

6.5. Future Directions

- Multi-Organizational Validation. Extend the validation to multiple organizations to determine whether the observed weekly invariance is universal in CI/CD or dependent on specific organizational context.
- Causal Transfer. Incorporate known events (holidays, product launches) as causal features, enabling explicit modeling of non-stationary periods.
- Online Adaptation. Implement incremental learning to adjust transferred patterns as the organization evolves (team growth, process changes).
- Cross-Platform Transfer. Validate the transfer of patterns from GitHub Actions to GitLab CI or Jenkins, enabling platform-agnostic autoscaling.

7. Conclusions

This paper presented a cross-temporal transfer learning methodology for CI/CD that resolves the heterogeneity problem between data sources from non-overlapping time periods. Our approach combines quantile mapping for distributional alignment with seasonality extraction and validation, producing a

target-domain dataset with realistic workload patterns without requiring shared temporal coverage between sources.

Key Contributions

1. Temporal Invariance Validation: we demonstrated that weekly organizational patterns in software development remain stable across quarters ($r = 0.8739$, $p = 0.0101$; confirmed at hourly granularity with $r = 0.742$, $p < 0.001$), establishing the theoretical basis for cross-temporal transfer.
2. Quantile Mapping Efficacy: we showed that distribution-preserving transformations (CV: 0.42 → 0.67, skewness: 0.18 → 1.21, kurtosis: 1.12 → 3.38) effectively align target-domain data with the real workload characteristics of the source domain, preserving distributional shape, variability, and extreme values.
3. Methodological Contribution: we established a reproducible framework – with an explicit validation criterion ($r > 0.7$) and a stage-wise ablation study – applicable to other CI/CD platforms and operational domains with stable organizational seasonality.

Contributions and Limitations by Dimension. To respond directly to each analyzed dimension of the study, Table 5 breaks down the specific contribution and the corresponding limitation identified for the four dimensions evaluated in this work (Table 7).

Table 7. Contributions and Limitations by Analyzed Dimension

Dimension	Contribution	Limitation
Temporal (seasonality) invariance	Demonstrated that weekly organizational activity patterns are stable across non-overlapping quarters, at both daily (r = 0.874) and hourly (r = 0.742) granularity.	Validated with n = 7 daily points; relies on the assumption of stable, synchronous (weekday-based) work schedules that may not hold for distributed or asynchronous teams.
Distributional alignment (quantile mapping)	Corrected CV, skewness, and kurtosis of target-domain durations to closely match the source domain, without assuming a Gaussian shape.	Requires enough data to estimate stable quantiles (n > 1000); Q1 source durations were reconstructed from aggregated billing minutes rather than measured directly.
Methodological framework	Established a reproducible, four-stage pipeline with an explicit acceptance criterion (r > 0.7) and a stage-wise ablation study isolating each stage's contribution.	Validated on a single source-target pair from one organization and one CI/CD platform (GitHub Actions); cross-platform and multi-organizational generalization remain untested.
Practical / deployment impact	Enables cold-start training-data generation and privacy-preserving reuse of external	Does not yet model non-stationary events (holidays, incidents, product launches) or organizational drift;

	public logs, reducing reliance on months of accumulated proprietary history.	continuous monitoring is recommended but not implemented here.

Practical Impact. This work demonstrates that human organizational behavior provides transferable structure across distinct time periods. By identifying and validating invariant patterns (weekly work cycles), we produce training datasets with high distributional and seasonal fidelity without relying on extensive historical data from the deployment period. The methodology is directly applicable to organizations that require representative datasets for new CI/CD infrastructure without sufficient history, migration between CI/CD platforms transferring learned seasonality patterns to a new context, academic research requiring realistic CI/CD datasets with privacy preservation, and any domain with stable organizational seasonality (DevOps, capacity management, resource planning).

Final Remark. Transferring patterns across temporal domains is feasible when an underlying invariant characteristic exists – in CI/CD, the human work calendar. By identifying, validating, and transferring this distributional and temporal invariance, we enable the use of heterogeneous data from non-overlapping periods as complementary sources of knowledge.

8. References

Beloglazov, A., Buyya, R., Lee, Y. C., & Zomaya, A. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. In M. V. Zelkowitz (Ed.), *Advances in Computers* (Vol. 82, pp. 47–111). Elsevier. https://doi.org/10.1016/B978-0-12-385512-1.00003-7

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. *Machine Learning, 79*(1-2), 151–175. https://doi.org/10.1007/s10994-009-5152-4

Bisong, E., Tran, E., & Baysal, O. (2017). Built to last or built too fast? Evaluating prediction models for build times. *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 487–496. https://doi.org/10.1109/MSR.2017.14

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control* (5th ed.). Wiley.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Calheiros, R. N., Masoumi, E., Ranjan, R., & Buyya, R. (2014). Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE Transactions on Cloud Computing, 3*(4), 449–458. https://doi.org/10.1109/TCC.2014.2350475

Cannon, A. J., Sobie, S. R., & Murdock, T. Q. (2015). Bias correction of GCM precipitation by quantile mapping: How well do methods preserve changes in quantiles and extremes? *Journal of Climate, 28*(17), 6938–6959. https://doi.org/10.1175/JCLI-D-14-00754.1

Chen, B., Chen, L., Zhang, C., & Peng, X. (2020). BuildFast: History-aware build outcome prediction for fast feedback and reduced cost in continuous integration. *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 42–53. https://doi.org/10.1145/3324884.3416616

Claes, M., Mäntylä, M. V., Kuutila, M., & Adams, B. (2018). Do programmers work at night or during the weekend? *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*, 705–715. https://doi.org/10.1145/3180155.3180193

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics, 6*(1), 3–73.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates.

Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM, 56*(2), 74–80. https://doi.org/10.1145/2408776.2408794

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts. https://otexts.com/fpp3/

Jin, X., Park, Y., Maddix, D., Wang, H., & Wang, Y. (2022). Domain adaptation for time series forecasting via attention sharing. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, PMLR 162, 10280–10297.

Kinsman, T., Wessel, M., Mens, T., & Serebrenik, A. (2022). How do software developers use GitHub Actions to automate their workflows? *Proceedings of the 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 420–431. https://doi.org/10.1109/MSR52588.2021.00054

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets)*, 50–56. https://doi.org/10.1145/3005745.3005750

Maraun, D. (2013). Bias correction, quantile mapping, and downscaling: Revisiting the inflation issue. *Journal of Climate, 26*(6), 2137–2143. https://doi.org/10.1175/JCLI-D-12-00821.1

Moriconi, S., Janes, A., & Russo, B. (2024). *GHALogs: A large-scale dataset of executed GitHub Actions workflows* [Data set]. Zenodo. https://doi.org/10.5281/zenodo.10813696

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Patki, N., Wedge, R., & Veeramachaneni, K. (2016). The synthetic data vault. *Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics*, 399–410. https://doi.org/10.1109/DSAA.2016.49

Pintye, P., Horváth, T., Balogh, A., & Charaf, H. (2024). Autoscaling GitHub Actions runners: Benefits and challenges. *Proceedings of the 2024 IEEE International Conference on Cloud Computing*, 145–152.

Rossi, F., Cardellini, V., & Presti, F. L. (2019). A review of autoscaling techniques for elastic applications in cloud environments. *Journal of Grid Computing, 17*(4), 699–726. https://doi.org/10.1007/s10723-019-09501-x

Saidani, I., Ouni, A., & Mkaouer, M. W. (2022). Improving the prediction of continuous integration build failures using deep learning. *Automated Software Engineering, 29*(1), 21. https://doi.org/10.1007/s10515-021-00319-5

Saxena, D., Kumar, J., Singh, A. K., & Schmid, S. (2023). Performance analysis of machine learning centered workload prediction models for cloud. arXiv preprint arXiv:2302.02452.

Subirats, J., & Guitart, J. (2015). Assessing and forecasting energy efficiency on cloud computing platforms. *Future Generation Computer Systems, 45*, 70–94. https://doi.org/10.1016/j.future.2014.11.008

United Nations. (2015). *Transforming our world: The 2030 agenda for sustainable development* (General Assembly resolution A/RES/70/1).

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., & Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in GitHub. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 805–816. https://doi.org/10.1145/2786805.2786850

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data, 3*(1), 1–40. https://doi.org/10.1186/s40537-016-0043-6

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A comprehensive survey on transfer learning. *Proceedings of the IEEE, 109*(1), 43–76. https://doi.org/10.1109/JPROC.2020.3004555