Enhancing Missing Data Imputation Through Network-Driven Latent Embeddings

24167356 [1]

Abstract

Missing data in time series is a pervasive problem that greatly affects the ability of advanced analysis. Presented with the `Louisiana Tank farm` dataset, we develop an advanced contrastive learning imputation model that imposes domain knowledge through the use of a graph regularization term. We compare our results against a Self-Attetion-Imputation Time Series (SAITS) model and an autoencoder. When evaluated over the full dataset, we achieve unbiased estimates (p=0.56) with an average improvement of 1.15% in RMSE, 5.20% in MAE, and 11.75% in MAPE over state-of-the-art models.

1. Introduction

Multivariate time series imputation remains a pivotal problem in advanced data analysis. Time series data is ubiquitous, stretching across healthcare, finance, meteorology, and, in this paper, oil tank data. However, recordings are not always complete: instrument malfunctions, data corruption, communication failures, weather restrictions, or even regulatory changes produce missing values. One widespread solution to this issue is imputation: the process of replacing missing points in a dataset based on observed values.

1.1. Background

The issue of missing values impairs the interpretability of data, potentially leading to reduced statistical power or biased results if not properly addressed. Traditional methods for solving this problem fall into two categories: deletion and imputation. The former can introduce a myriad of additional issues. For example, although list-wise deletion preserves data integrity, it often loses substantial data in the process, as suggested by Ndifon (2023). Pairwise deletions can rectify this by preserving data, but, as Graham (2009) shows, this can produce inconsistencies and biased estimates. Therefore, there are two clear advantages to imputation over deletion: i) partially observed data may still be helpful and informative for advanced analysis and ii) deletion introduces bias, while correctly specified imputation estimates are unbiased (White and Carlin, 2010).

However, imputation becomes increasingly difficult with higher levels of missingness (Wi et al., 2024). Despite being efficient (in terms of latency), simple imputation methods, such as mean, median, or mode imputation, ignore relationships with other variables, can underestimate variability, and can distort data distributions. More sophisticated methods, like regression imputation or multiple imputation in (Nijman et al., 2021), can improve upon these methods, but in relying on linear regression models, they can be sensitive to outliers and potentially lead to biased imputations (Schwerter et al., 2024).

In contrast, deep learning approaches, are designed to model intricate patterns, making them more adept at handling the complexities inherent in multivariate time series data. For example, autoencoders, are designed to learn a representation of a dataset from the input layer and attempt to reproduce it at the output layer (Wang et al., 2023). Similarly, advanced methods like the Self-Attention-based imputation for time series (SAITS) model proposed by Du et al. (2023), use self-attention mechanisms for multivariate time series imputation.

Despite the success of these models, they are not without limitations. Latent space assumptions in autoencoders may not match real data (Jiao et al., 2023). Furthermore, standard autoencoders fail to generalize well in imputation tasks(Gondara and Wang, 2018) , while overcomplete networks require extensive resources. Moreover, the added complexity from attention in SAITS does not guarantee benefits (Fang et al., 2023). Requiring more extensive resources, it may not capture the intricate interdependencies between different variables over time as comprehensively as desired, potentially affecting imputation accuracy.

As an alternative approach, Shi et al. (2023) turns towards contrastive learning for difficult imputation tasks. Contrastive learning creates discriminative latent embeddings, allowing for simple imputation models to be used following it. However, Pan et al. (2025) notes that existing graph contrastive learning methods struggle to capture local spatial and temporal domains, without explicit inclusion. The lat-

ter observation catalyzed our use of a contrastive learning model paired with a graph regularizer that was constructed from spatiotemporal dependencies.

1.2. Motivation

Aligning with the insight from Pan et al. (2025), Lee et al. (2024) isolates a "class collision" problem in contrastive learning, where negative samples may belong to the same class. They, too, introduce consistency regularization to pull semantically similar nodes closer in the embedding space. We leverage this by using graph regularization to ensure that spatially and temporally proximate tanks, even if treated as negatives, are appropriately aligned in the latent space, enhancing the quality of the embeddings for imputation.

Our initial analysis of the `Louisiana Tank farm` dataset signified missingness as a pivotal issue, with tank data being reported infrequently. We compare simple methods for imputation, such as Mean, Median, Mode, Linear Interpolation, and KNN, to existing advanced methods, such as the SAITS and autoencoder models, which provide a baseline for our model.

We choose contrastive learning as it ensures that the embeddings are robust by maximizing agreement between augmented views of the same sample, making the model less sensitive to noise and in our case, the issue of missingness. This proposed solution helps us investigate our provocative question: *Under circumstances of severe missingness, will added graph regularization aid a contrastive learning model to outperform other advanced imputation models in terms of RMSE, MAE, MAPE, and R^2?*

2. Dataset Challenges

2.1. Bias & Structure

The `Louisiana Tank Farm Dataset` comprises 44,685 entries from 2014-01-18 to 2018-04-05, covering 693 unique oil tanks across Louisiana. For each measurement, the dataset includes the following features for each `tank_id`: `fill_pct`, calculated as total_volume/max_vol, the timestamp of the measurement (`imaging_time`), the type of satellite used to capture the image, the tank's location in longitude and latitude, and the tank farm id, which identifies the tank farm containing the tank.

Initial plotting of the 28 tank farm locations, as shown in Figure 1, illustrates a strong concentration of farms in the Baton Rouge area, extending down to New Orleans. The remaining tank farms are more dispersed, located in areas such as Lafayette, Lake Charles, and Shreveport. This dense clustering of tank farms in certain regions suggests

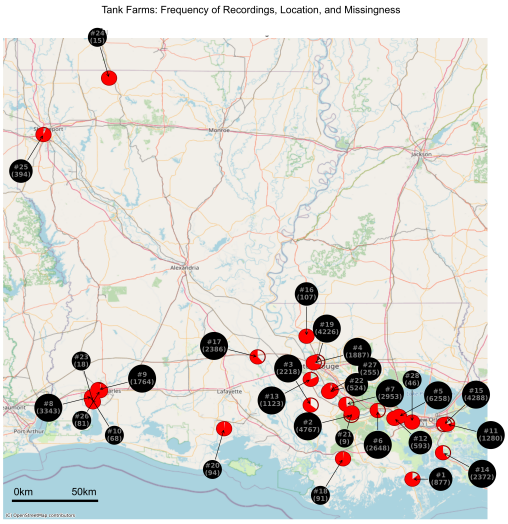

Figure 1: Rank of each tank farm by maximum tank volume (e.g., #1, #2), the number of recordings per tank (e.g., (123)), and a red circle scaled by the number of recordings relative to the tank farm with the most recordings.

potential spatial dependencies, as tanks in close proximity might have correlated `fill_pct` values due to shared operational or environmental factors. This observation motivated the exploration of graph networks to model these dependencies (Section 5).

In Figure 2, the initial distribution of `fill_pct` (left) is broad, with fewer tanks at near-full (fill_pct > 0.9) or near-empty (fill_pct < 0.1) levels. The density curve exhibits clustered peaks, particularly between 0.2 and 0.8, suggesting that operational practices may favor maintaining tanks at specific fill levels, possibly for logistical efficiency.

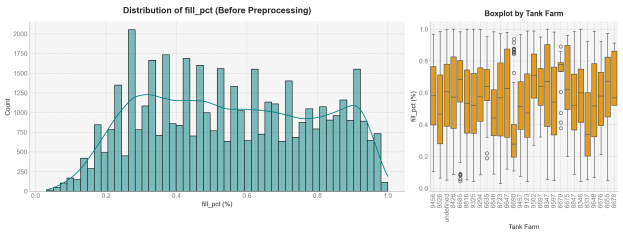

Figure 2: Distribution of fill_pct (left) and boxplot of fill_pct for each tank farm (right) before restructuring.

The box plot (right) reveals variation in `fill_pct` spread across tank farms, indicating that different farms maintain distinct median fill levels. For example, tank farms 6685 and 8426 have tight interquartile ranges, suggesting sta-

ble operational patterns, while others exhibit wider spreads or numerous outliers, indicating more variability. The box plot also includes an "undefined" tank farm, which, while unusual, is not a primary concern of our research.

To examine temporal patterns in recordings, we plotted the number of satellite images taken per month (Figure 3) and per year (Figure 4), given the time series nature of the dataset.

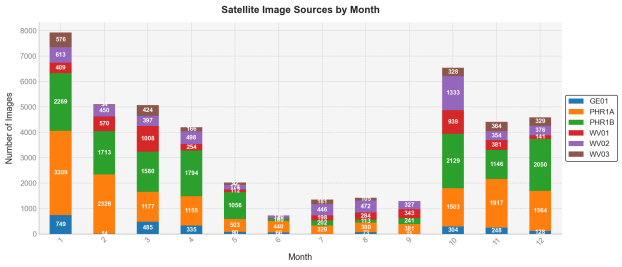

Figure 3: Satellite images separated by monthly occurrence.

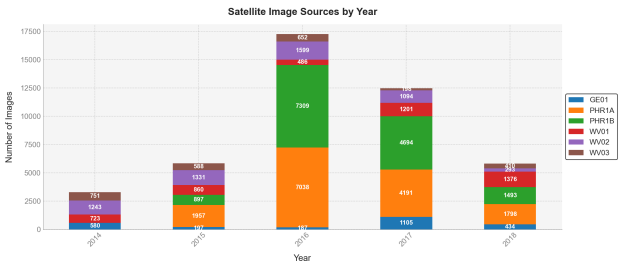

Figure 4: Satellite images separated by yearly occurrence.

These Figures highlight significant inconsistencies in measurement frequency. Winter months were disproportionately favored for image capturing, accounting for approximately 39.44% of all recordings, compared to 7.86% in Summer. Additionally, recordings peaked in 2016 with 38.65% of total images, while 7.38% of images were taken in 2014, 13.05% in 2015, 27.94% in 2017 and 12.99% in 2018. These temporal biases——across seasons and years——were not apparent when simply counting null rows. They also suggest that the `fill_pct` distribution in Figure 2 may be unrepresentative, as the overrepresentation of Winter months and certain years could skew the distribution toward values typical of those periods.

2.2. Primary Issue

To understand the impact of recording inconsistencies, we plotted the mean `fill_pct` over time for a subset of tank farms in Figure 5. For brevity, we display only 6 of the 28 unique tank farms, but the figure clearly shows large gaps in recordings.

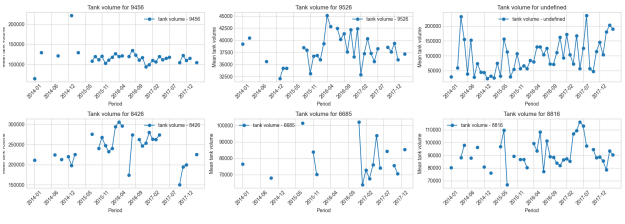

Figure 5: Mean fill_pct over time for a subset of tank farms.

Consecutive measurements for each tank_id were captured 28.29 days apart on average, with a median interval of 20.85 days. The minimum interval was 5.25 days, and the maximum was 1,097 days (approximately 3 years). Due to the right-skewed distribution of these intervals, the interquartile range (IQR) provides a more robust measurement of recording frequency, spanning from 18.22 days to 24.93 days.

To investigate the missingness in `fill_pct`, we first augmented the dataset with additional features, such as weather and spatial data (Section 3.2), and then conducted statistical tests to determine the missingness mechanism. To create a complete panel dataset, we added rows for every `tank_id` and month combination between 2014-01-18 and 2018-04-05, setting dynamic features like `fill_pct` to NaN when no measurement was available, while retaining static features like `max_vol`. We chose to reconstruct the dataset on a monthly frequency because the median interval between measurements (20.85 days) is close to one month. A daily frequency resulted in over 99% missingness, making imputation infeasible. At the monthly level, the dataset contained 18,434 missing `fill_pct` values out of 36,036 total rows, yielding an overall missingness rate of 51.15%.

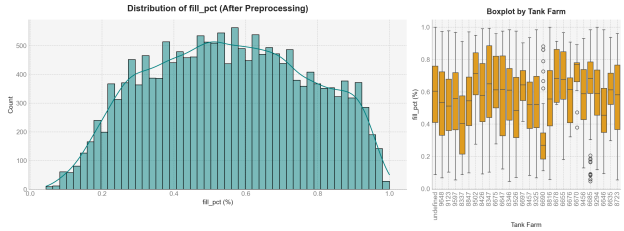

Figure 6: Distribution of fill_pct (left) and boxplot of fill_pct for each tank farm (right) after restructuring.

This restructuring reduced the dataset from 44,685 rows to 36,036 by averaging multiple `fill_pct` measurements within the same month for a given `tank_id`, creating a feasible yet challenging imputation task. We focus on imputing `fill_pct` because it is a standardized value, and the `total_volume` can be derived from it using the known constant `max_vol`.

After restructuring, the distribution of `fill_pct` (Figure 6, left) became more concentrated between 0.4 and 0.9, with a smoother, bell-shaped curve. This change resulted from averaging recordings within each month and adding NaN values for months with no recordings, which increases the proportion of missing values but better reflects the true data structure.

We then determined if the probability of a recording being missing was the same for all cases, causing the data to be labeled as missing completely at random (MCAR). This implies that causes of the missing data are unrelated to the known data. Instead, we aimed to conclude the data was missing at random (MAR), where missingness depends on observed variables, such as season (e.g., hot weather) or WTI prices, rather than the missing values themselves

Therefore, we initially conduct Welch's t-tests to compare the means of each feature (season, WTI prices, etc.) between groups with missing and non-missing `fill_pct` values, assuming unequal variances. Despite the normality assumption, the large sample size ($n > 1000$) ensures robustness through the Central Limit Theorem

Test	Statistic	p-value	d.o.f.
MCAR Violation	$t \in [-45.6, 62.2]^*$	$\forall f \in \mathcal{F},\ p_f < 0.05$	–
Season (MAR)	$\chi^2 = 1697.48$	< 0.001	3
Tank ID (MAR)	$\chi^2 = 3108.91$	< 0.001	692

Table 1: Missingness mechanism tests. MCAR rejected for all features ($|t|_{\text{range}} = 0.85$–$62.2$).

We find significant evidence against MCAR. The MCAR t-test compares the mean of other features between missing and non-missing `fill_pct` groups. All features show significant differences ($p < 0.05$) between missing/non-missing groups. This indicates the missingness present in `fill_pct` is not MCAR, pointing towards MAR. If missingness were MCAR, the means of other features should be similar between missing and non-missing groups of `fill_pct`.

Subsequently, our chi-square test for missingness of `fill_pct` and season returned a value of 1697.48 with a $p_{value} < 0.05$. This confirms that missingness in `fill_pct` depends on season, supporting the MAR hypothesis. This matches the seasonal variation seen in Figure 3. We notice significantly high missingness in the summer, potentially due to issues with cloud coverage or sun distortions in image capturing, and the lowest missingness in the winter, aligning with reporting periods and superior weather periods for satellite imaging.

A significant p-value from the chi-square test between missingness of `fill_pct` and `tank_id` (Table 1) indicates that certain tanks are more likely to have missing

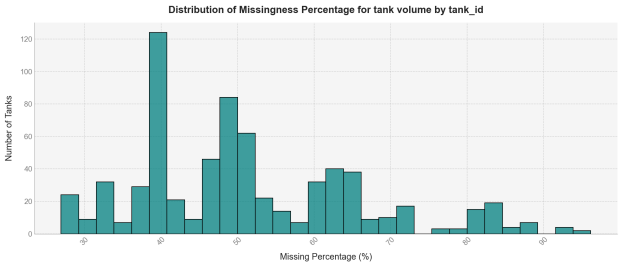

Figure 7: Percentage of missing values for tank volume across tank_ids.

`fill_pct` values than others, rejecting MCAR (Figure 7. Missingness rates vary substantially across tanks, ranging from 38.5% to 69.2%. This pattern, combined with the association of missingness with observed variables (`season` and `tank_id`), supports MAR. While MNAR cannot be ruled out——since missingness could theoretically depend on unobserved `fill_pct` values——we assume MAR is plausible, as the observed variables sufficiently explain the missingness without evidence of an MNAR mechanism.

Both Welch's t-tests and chi-square tests assume independent observations, an assumption violated due to autocorrelation (Section 2.3). This may affect p-values, potentially overestimating significance, but the strong results ($p < 0.001$) suggest this has minimal impact.

This analysis highlights missingness as the central challenge of our dataset. Without accurate imputation, downstream analytical tasks become unreliable or infeasible. This motivates our focus on developing an advanced imputation method that effectively integrates domain knowledge into the learning process.

We utilize the missingness metrics identified above to determine which values to mask when training our imputation methods. Previously, we had applied masking at random, but this did not accurately represent the MAR nature of our dataset. Therefore, we incorporate the computed probability of missingness for each season and tank, and use these to mask the test set in a stratified way. This allows us to mimic the real missingness pattern in the dataset, and scale these probabilities to achieve a target missingness level applied to all methods.

2.3. Spatial & Temporal Dependencies

The high level of missingness posed challenges for our initial autocorrelation analysis, which requires consecutive observations to compute lagged correlations accurately. When gaps are present, they are treated as a continuous series, distorting the temporal structure. For example, if $\texttt{fill\_pct}_t$ is missing, $\texttt{fill\_pct}_{t-1}$ and $\texttt{fill\_pct}_{t+1}$ are paired, leading to the computation of a lag-2 correlation

instead of a lag-1 correlation. We noticed these large gaps resulted in an underestimation of the true autocorrelation.

Without addressing missingness, a preliminary autocorrelation analysis of lag 1 across tanks yielded a mean autocorrelation of 0.15, with t-test confirming its significance ($p < 0.05$). However, this value fell short of the 95% confidence interval, suggesting that the temporal dependencies may be underestimated due to missing data.

To resolve this, we elected to impute the missing values before conducting a comprehensive autocorrelation analysis. The best "simple" imputation method was determined based on the combined score between RMSE, MAE, MAPE, and R^2 for `fill_pct`. We compared several methods, including Mean, Median, Mode, Forward Fill, Linear Interpolation, Iterative, and KNN imputation, with the results in Section 3.5. Linear interpolation returned the lowest RMSE, highest R^2, and the greatest performance, leading to its use throughout this section.

After imputation, we mitigated several autocorrelation analysis issues. First, we addressed the risk of inflated autocorrelation values due to non-stationary trends. Second, we ensured that tanks with very fewer than 10 observations were excluded to avoid unreliable estimates that could lead to spurious significance in our t-test. Finally, we accounted for multiple testing across lags by applying a Bonferroni correction, as we explore temporal dependencies up to a lag of 3.

To assess non-stationarity, we used the Augmented Dickey-Fuller (ADF) test, supplemented by the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test to address ADF's low power against near-unit-root processes. Before imputation, 44.57% of 693 tank-specific `fill_pct` series were non-stationary (ADF$p > 0.05$ and/or KPSS$p < 0.05$). After linear interpolation, this rose to 74.39%, likely due to introduced trends, prompting first-order differencing ($\Delta\texttt{fill\_pct}_t = \texttt{fill\_pct}_t - \texttt{fill\_pct}_{t-1}$) before autocorrelation analysis

The results of the autocorrelation analysis on the differenced series are presented in Figure 8. Of 693 tanks with sufficient data, 472 (68.1%) exhibited a lag-1 autocorrelation exceeding the 95% significance threshold ($= 0.2718$). The mean autocorrelation at lag 1 was -0.3771, with a Bonferroni-adjusted p-value < 0.001, from a one-sample t-test:

$$t = \frac{\bar{r} - 0}{s/\sqrt{n}}$$

where $\bar{r} = -0.3771$ is the mean autocorrelation for lag 1, s is the standard deviation of lag-1 autocorrelations across tanks, n is the number of tanks, and H_0 is the null hypothesis that the true mean autocorrelation is 0.

A two-tailed permutation test robust to the non-normal distribution of autocorrelations (Section 3.2), yielded $p_{perm} < 0.001$, with no permuted means as extreme as -0.3771, reinforcing the t-test result. At lag 2, the mean autocorrelation was -0.0477, with an adjusted p-value < 0.01, indicating it was still significant but weaker. At lag 3, the mean autocorrelation was -0.0198 (adjusted p-value = 0.06), indicating that temporal dependencies diminish after a few lags.

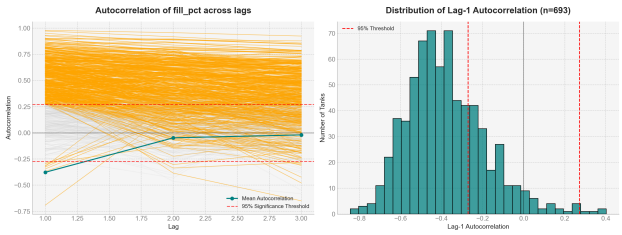

Figure 8: Autocorrelation of the differenced `fill_pct` series from lag 1 to 3 (left) and the distribution of lag-1 autocorrelations across tanks (right).

The distribution of lag-1 autocorrelations, shown in the right panel of Figure 8, is centered around -0.4, with most values ranging between -0.8 and 0.2. The 95% significance threshold highlights that 68.1% of tanks have significant lag-1 autocorrelations, reinforcing the prevalence of temporal dependencies across the dataset and supporting our use of a graph regularization model (Section 5).

To explore spatial dependencies in `fill_pct` values, we computed the Spearman correlation between `distance_to_nearest_tank_id` and absolute differences in detrended `fill_pct`, yielding -0.0033 (Bonferroni-adjusted p = 0.7281), with a permutation test (100 iterations, p = 1.0) confirming no monotonic relationship. However, mutual information (Section 3.6) revealed a significant non-linear dependency ($p < 0.05$), justifying spatial modeling in Section 5.

3. Feature Selection

3.1. Additional Features

To enhance imputation, we augmented the dataset with features capturing additional context for missing data patterns. Weather data (temperature, cloud cover, precipitation) was added to reflect environmental conditions potentially affecting imaging and `fill_pct` similarity across tanks, sourced from nearby stations using the Meteostat API. Temporal features (day of week, season) were included to model periodic imaging biases (pre-noon or non-summer prevalence), while EIA reporting dates provided operational cycle context, all aimed at improving embeddings for similarity learning.

Spatial features, such as distance to the nearest port and

population density, were incorporated to refine graph construction and capture operational similarities. For example, tanks near ports may share fill patterns. Additionally, WTI and Brent Crude oil prices were considered for economic context.

Consequently, with 11 initial dynamic features and 2 static (`distance_to_nearest_port`, `distance_to_nearest_tank_id`) we added 3 lagged features (33 total) and 2 rolling statistics (22 total) for each of the dynamic features. This brought the total feature set to 68, and with `fill_pct` it reached 69. These were then evaluated using correlation and mutual information to select the most informative subset for the imputation task (Section 3.2) and 3.6).

3.2. Correlation

As our multivariate time-series dataset exhibits temporal autocorrelation within tanks and spatial correlation across regions, it violates the independence and normality assumptions of standard correlation and mutual information (MI) methods. Therefore, the following adjustments were critical for robust feature selection in our downstream imputation task.

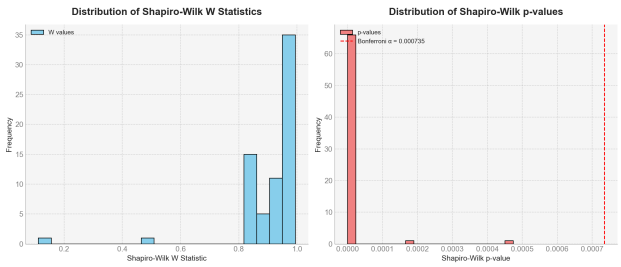

Figure 9: Distribution of Shapiro-Wilk test statistics (left) and p-values (right) for 69 features, with a Bonferroni-corrected threshold of $\alpha' = 0.000725$.

We assessed feature distributions using the Shapiro-Wilk test (Figure 9) on samples of 1,000 observations, revealing non-normality across all features (e.g., $p < 0.000725$ for `fill_pct` after Bonferroni correction). We selected the Shapiro-Wilk test for its higher statistical power in detecting non-normality in moderate sample sizes compared to tests like the Kolmogorov-Smirnov or Anderson-Darling tests (Razali and Wah, 2011). Despite W test statistics clustering near 1, all p-values are below the Bonferroni threshold, meaning all features are flagged as significantly non-normal.

However, because the test's sensitivity increases with sample size—potentially flagging trivial deviations—we paired it with Q-Q plots to contextualize deviations. We chose Q-Q plots over P-P plots because they directly compare sample and theoretical quantiles, thereby highlighting differences in the tails. The `fill_pct` distribution, for example, has lighter tails than a Gaussian, as shown in Figure 10. The Bonferroni correction in Figure 9 adjusts for multiple testing across the 69 features, controlling the family-wise error rate, with further discussion in Section 3.5.

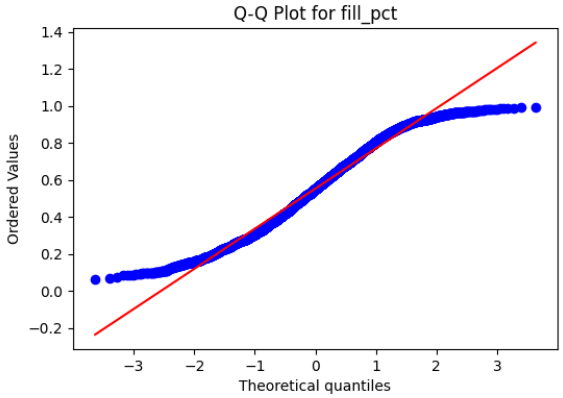

Figure 10: Q-Q plot of `fill_pct`, showing lighter tails compared to a Gaussian distribution.

Given its lighter tails (fewer extreme values), modeling `fill_pct` as a Lévy-alpha distribution is inappropriate, and standard transformations—log, square root, Box-Cox, arcsine—failed to normalize the data consistently across features. For instance, log-transformations of `fill_pct` yielded $p < 0.05$ through the Shapiro-Wilk tests in initial analyses, so we retained untransformed data.

Given Pearson correlation's normality assumption (Schober et al., 2018), we adopted Spearman correlation, a non-parametric method based on ranks:

$$\rho = \text{Corr}(\text{Rank(X)}, \text{Rank(Y)}) = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)},$$

where d_i is the rank difference between paired observations, and n is the sample size. Weighted adjustments were applied using tank farm inverse counts ($= 1/\text{frequency of tank farms}$) to balance the influence of tank farms with varying observation counts, handling non-normality but assuming monotonicity.

Standard correlation assumes i.i.d. observations, which our autocorrelated data violates. Random bootstrapping risked forward bias by disrupting temporal structure, yielding overly optimistic intervals. Instead, we used `TimeSeriesSplit` with 5 folds to preserve chronological order. For observations $T_1, T_2, \ldots, T_n$, we defined split indices $t_0 = 1, t_1, t_2, t_3, t_4, t_5 = n$, where fold i (for $i = 1, \ldots, 4$) has:

$$\text{Train set: } \{T_{t_0}, \ldots, T_{t_i}\}, \quad \text{Test set: } \{T_{t_i+1}, \ldots, T_{t_{i+1}}\}.$$

Correlations were computed prior to imputation, allowing us to compute true relationships among features. Imputation could inflate our correlations by imposing structure, misleading out feature selection (van Buuren, 2018). Correlations were calculated over training folds and averaged, reducing data leakage. To embed temporal dynamics, we added lagged features ($X_{t,\text{lag}k} = X_{t-k}$) and 7-day rolling means ($X_{t,\text{roll mean}} = \frac{1}{7}\sum_{j=t-6}^{t} X_j$) for dynamic features, mitigating i.i.d. violations explicitly.

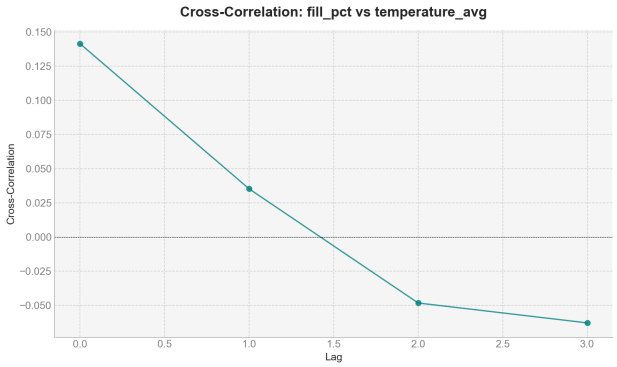

Figure 11: Cross correlation of fill_pct and temperature_avg over different lags.

Cross-correlations identified optimal lags, guiding the decision of which lag to include for each feature. The `temperature_avg` feature, for example, clearly shows in Figure 11 that lag 1 resulted in the highest cross-correlation value, which led to its inclusion in further testing. Additionally, embedding temporal dynamics through lagged and rolling features aligns with existing work (Hyndman and Athanasopoulos, 2021).

3.3. Permutation Tests

To assess the significance of Spearman correlations amidst non-normal, temporally dependent data, we used permutation tests, a distribution-free approach robust to these violations (Good, 2005).

For an observed correlation ρ_{obs}, permutation tests evaluate the null hypothesis of no association by assuming one variable's values (e.g., `wti_price`) are independent of the other (`fill_pct`). We shuffled one variable's values 100 times ($N = 100$), recomputing the weighted Spearman correlation $\rho_{\text{perm},i}$ each time:

$$p = \frac{1}{N}\sum_{i=1}^{N} I(|\rho_{\text{perm},i}| \geq |\rho_{\text{obs}}|),$$

where $I(\cdot)$ is 1 if true and 0 otherwise. This p-value estimates the proportion of permuted correlations as extreme as ρ_{obs}, offering an empirical significance measure.

3.4. Bootstrapping for Confidence Intervals

Permutation tests yield p-values but not correlation variability. We thus applied bootstrapping within each `TimeSeriesSplit` fold, resampling with replacement to estimate 95% confidence intervals. For each fold, we generated $B = 20$ bootstrap samples of size equal to the fold's sample size n, computing the weighted Spearman correlation ρ_b for each:

$$\{\rho_b\}_{b=1}^{B},$$

with the CI as:

$$\text{CI} = [\text{2.5th percentile}, \text{97.5th percentile}] \text{ of } \{\rho_b\}_{b=1}^{B}.$$

This range estimates the true correlation's likely bounds. We chose $B = 20$ to balance precision and computational feasibility, as higher values (e.g., 100) were impractical on our hardware (MacBook Air, 1.2 GHz Quad-Core Intel® Core™ i7).

Resampling within folds preserves temporal structure, and we then averaged confidence intervals across folds.

3.5. Multicollinearity Assessment with VIF

After correlation analysis, we tackled multicollinearity, which can inflate variance and obscure feature contributions in imputation and modeling. First, we visualized this issue by creating clusters of similar or redundant features using a dendrogram. This dendrogram was generated using Spearman correlations—computed as one minus the absolute correlation—to form a distance matrix, and average linkage was employed to merge features into clusters.

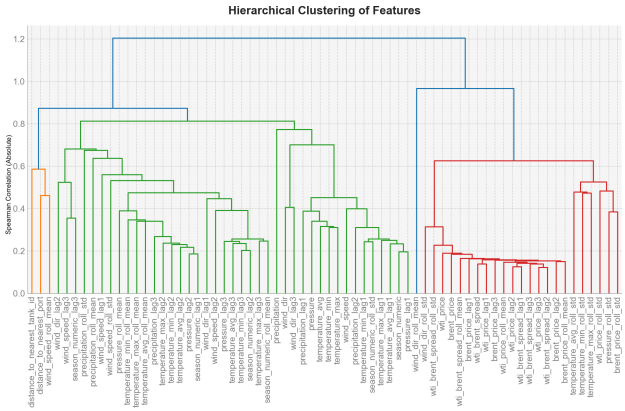

Figure 12: Dendrogram of similarity.

Subsequently, we quantified highly correlated features—especially lagged and rolling derivatives (e.g., `wti_price_lag1`, `wti_price_roll_mean`)—using the Variance Inflation Factor (VIF):

$$\text{VIF}_j = \frac{1}{1 - R_j^2},$$

where R_j^2 is the coefficient of determination from regressing feature j (e.g., `wti_price`) on all others. A VIF $>$ 40 indicates significant redundancy—a threshold selected to balance the retention of useful time series predictors against the removal of excessive collinearity due to numerous lagged and rolling features.

Lagged features from the same base variable (e.g., `wti_price_lag1`, `wti_price_lag2`) are naturally collinear due to temporal autocorrelation (Section 2.3). To preserve time-dependent relationships, we computed VIF on base features and assigned these values to their derivatives, balancing multicollinearity reduction with temporal structure retention.

VIF requires complete data, but dropping NaNs removed 18,434 rows due to pervasive missingness, an impractical solution. This created a circular dependency: feature selection needed imputation, yet imputation required selected features. Therefore, we i) computed Spearman correlations pairwise on available data, preserving true relationships without imputation bias (e.g. KNN assumes row similarity and mean imputation assumes randomness). Next, we ii) imputed missing values with Linear Interpolation (Table 2) for VIF, chosen for its superior performance across RMSE, MAE, MAPE, and R^2, followed by KNN (k=5) for final VIF computation. We intentionally did not use one of our 3 advanced methods, in order to avoid biasing our final imputation comparison.

With $m = \frac{k(k-1)}{2}$ pairwise comparisons ($k = 69$, $m = 2346$), the family-wise error rate increases. Bonferroni correction adjusts the significance level to $\alpha' = \frac{0.05}{2346} \approx 2.13 \times 10^{-5}$, controlling Type I errors but conservatively, potentially overlooking weaker effects. This adjusted threshold was used in correlation heatmaps to identify significant relationships.

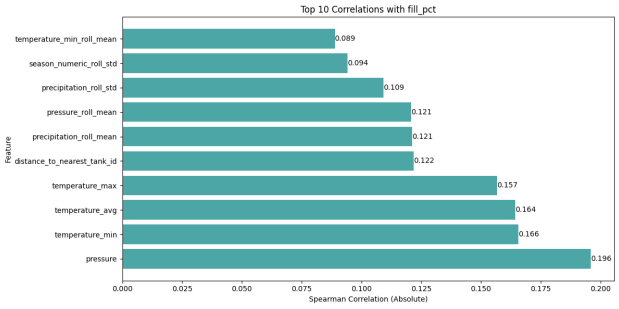

Figure 13: Top 10 features by Spearman correlation with `fill_pct` post-cross-correlation analysis.

3.6. Mutual Information and i.i.d. Mitigation

Mutual Information (MI) complements `corr.py`'s correlation analysis by capturing non-linear and non-monotonic dependencies between features and `fill_pct` for feature selection. MI quantifies how much knowing X (e.g., `wti_price`) reduces uncertainty about Y (`fill_pct`) and can be expressed as the Kullback-Leibler Divergence (KLD) between the joint and product of marginal distributions:

$$I(X;Y) = D_{KL}(p(x,y) \| p(x)p(y)) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}, \quad (1)$$

where $p(x,y)$ is the joint probability and $p(x), p(y)$ are marginals. We used `mutual_info_regression`'s k-NN approximation (Kraskov et al., 2004) to implicitly estimate the required probability densities by analyzing the local neighborhood of each observation, thus avoiding explicit distribution estimation (Section 4.2). However, the time-series nature of our data posed further challenges.

The i.i.d. assumption of MI was violated by autocorrelation within `tank_id` observations (Section 2.3), with 68.1% of tanks showing significant lag-1 autocorrelation (mean $= -0.3771$, $p < 0.001$), inflating estimates by reducing effective sample size. Additionally, non-linear spatial dependencies were present, as evidenced by the significant mutual information between `fill_pct` and `distance_to_nearest_tank_id` (MI $= 0.104$, $p < 0.05$).

Initially, random bootstrapping ($B = 50$) disrupted temporal order, invalidating MI for time-series use. In our mutual information analysis, we replaced bootstrapping with `TimeSeriesSplit` (5 folds), computing MI on training folds and averaging to mitigate variance inflation. Lagged features and rolling statistics embedded temporal dependencies, easing MI's burden. We tuned the number of nearest neighbors k in $[3, 5, 10]$ using 3-fold cross-validation, selecting the value that maximized MI.

While Section 3.2 identified monotonic relationships with `fill_pct` through cross-correlation, Spearman, and VIF clustering, MI complemented these results and revealed non-linear contributors that Spearman correlation alone could have missed.

We extracted the top 50% of features by MI, retaining those that also appeared in the recommended features from `corr.py`'s `keep_features`, thereby cross-validating their relevance. Therefore, our final dataset included the following features, alongside the target variable of `fill_pct`: `temperature_min`, `precipitation`, `wind_dir`, `precipitation_roll_mean`, `temperature_avg_roll_std`, `distance_to_nearest_port`, `distance_to_nearest_tank_id`, `season_numeric`, `wti_price`, and `time_diff`.

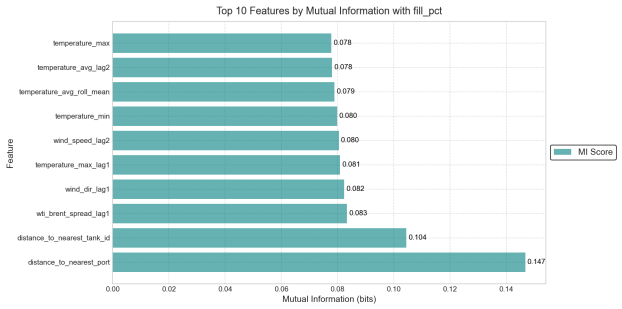

Figure 14: Top 10 features by Mutual Information with `fill_pct`.

4. Methodology

4.1. Simple Imputation Techniques

We implemented simple imputation methods to establish baselines for missing value recovery in `fill_pct`, using the same temporal dataset as our advanced models. Both simple and advanced approaches introduced artificial missingness of 20% to assess performance against known values. Evaluation employs walk-forward cross-validation with five folds, preserving temporal order, and uses RMSE, MAE, MAPE, R^2, and a paired t-test for bias (`ttest_rel`) on artificially masked points.

Method	RMSE	MAE	MAPE (%)	R^2	Bias p-value
Linear Interpolation	0.904	0.670	221.94	0.166	0.362
Mean	0.993	0.840	104.689	-0.007	0.373
KNN	0.993	0.840	104.689	-0.007	0.373
Iterative	0.993	0.840	104.858	-0.007	0.433
Median	0.996	0.842	108.474	-0.012	0.154
ForwardFill	1.022	0.752	282.981	-0.069	0.524

Table 2: Comparison of imputation methods, evaluated by masking known values and computing metrics via 5-fold cross-validation.

The simple methods of mean, median, forward fill, linear interpolation, KNN, and iterative imputation were applied directly without extensive training. For KNN, we normalized features using `StandardScaler`, which was fit on training data only, to ensure equal contribution to the distance metric, tuning the number of neighbors (k) through cross-validation to optimize RMSE ($k = 5$ was selected). As shown in Figure 2, Linear Interpolation had the lowest RMSE and the highest R^2. This resulted in its use to solve the circular problem we experienced in Section 3.5.

4.2. KNN

To use the K-nearest neighbors algorithm for imputation, we first defined a distance measure. Although many metrics—such as Manhattan or Minkowski distances—are available, we opted for the Euclidean distance due to its reported computational efficiency (Amirteimoori and Kordrostami, 2010), which is critical for minimizing runtime and resource usage on our simple device (Appendix B). For two data points

$$\mathbf{X} = (x_1, x_2, \ldots, x_n) \quad \text{and} \quad \mathbf{Y} = (y_1, y_2, \ldots, y_n)$$

the Euclidean distance is defined as:

$$d(\mathbf{X}, \mathbf{Y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Here, the squared terms (exponent $k = 2$) denotes the Euclidean distance, which is distinct from the hyperparameter k in k-NN.

The k in KNN denotes the number of nearest neighbors to consider. It is a hyperparameter that we set based on cross-validation. In our experiments, the lowest RMSE was achieved with $k = 5$. For determining k we only masked `fill_pct`, as opposed to all features, when conducting imputation.

Using KNN regression, the missing value for a query point is predicted as the weighted average of the values of its k nearest neighbors. In order to give closer neighbors more influence, we use an inverse-distance weighting scheme. That is, the weight w_i for the ith neighbor is defined as:

$$w_i = \frac{1}{d(\mathbf{X}, \mathbf{Y}_i)}$$

where $d(\mathbf{X}, \mathbf{Y}_i)$ is the distance between the query point $\mathbf{X}$ and its ith neighbor $\mathbf{Y}_i$.

4.3. Autoencoder

To provide a rigorous baseline for our contrastive learning and graph regularization (CL×GR) model we first implemented an autoencoder. It was chosen due to its wide applicability, simplicity, and, in particular, its strength in capturing feature relationships in latent space.

The autoencoder is a neural network that learns to compress (encode) its input into a lower-dimensional latent space and then reconstruct (decode) the input from that representation. This is useful in our imputation task, as the overall goal is to minimize the reconstruction error between the input and the output.

The vectorized input data, $\mathbf{x} \in \mathbb{R}^d$, where d is the dimensionality of the input, is transformed into a latent representation $\mathbf{z} \in \mathbb{R}^m$ (with $m < d$) using the following function:

$$\mathbf{z} = f(\mathbf{x}) = \sigma\left(\mathbf{W}_e \mathbf{x} + \mathbf{b}_e\right)$$

where $\mathbf{W}_e \in \mathbb{R}^{m \times d}$ is the weight matrix of the encoder, $\mathbf{b}_e \in \mathbb{R}^m$ is the bias vector, and $\sigma(\cdot)$ is a non-linear activation function, which we chose as the Rectified Linear Unit (ReLU).

The decoder reconstructs the original input, following this transformation, from the latent representation $\mathbf{z}$ using a linear transformation:

$$\hat{\mathbf{x}} = g(\mathbf{z}) = \mathbf{W}_d\mathbf{z} + \mathbf{b}_d$$

where $\mathbf{W}_d \in \mathbb{R}^{d\times m}$ is the weight matrix of the decoder and $\mathbf{b}_d \in \mathbb{R}^d$ is the bias vector.

To do this, the model is trained to minimize the *reconstruction error*, which we measure by the Mean Squared Error (MSE). For a single input $\mathbf{x}$ with reconstruction $\hat{\mathbf{x}}$, the loss is defined as:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 = \sum_{i=1}^{d}(x_i - \hat{x}_i)^2$$

For a dataset with N samples, the training objective becomes:

$$\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}^{(i)} - g(f(\mathbf{x}^{(i)}))\|_2^2,$$

where θ collectively represents the parameters $\{\mathbf{W}_e, \mathbf{b}_e, \mathbf{W}_d, \mathbf{b}_d\}$

Lastly, we use the Adaptive Momentum Estimation (Adam) optimization algorithm—a combination of momentum and adaptive learning rates—as it adjusts the learning rate for each parameter individually, reaching faster convergence under the resource constraints of our MacBook Air.

4.4. SAITS

The second baseline model we utilized was the Self-Attention Imputation for Time Series (SAITS) model. Its transformer architecture captures long-range dependencies through self-attention. While it outperforms simpler models, it inherently ignores *spatial relationships* between tanks—a limitation our graph-based approach addresses. By including SAITS, we ensure that any performance gains from our model stem from the explicit incorporation of graph structure, rather than merely superior temporal modeling.

Representing the input data as a sequence $\mathbf{X} \in \mathbb{R}^{T\times d}$, where T is the number of time steps and d is the number of features, the first step of the SAITS model is to transform the input data into a higher-dimensional embedding space:

$$\mathbf{X}_{\text{embed}} = \mathbf{W}_{\text{in}}\mathbf{X} + \mathbf{b}_{\text{in}}$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{d_{\text{model}}\times d}$ is the weight matrix, $\mathbf{b}_{\text{in}} \in \mathbb{R}^{d_{\text{model}}}$ is the bias vector, and d_{model} is the dimension of the model's hidden space.

The embedded input $\mathbf{X}_{\text{embed}}$ is then passed through a transformer encoder. The encoder consists of L identical layers, each containing a multi-head self-attention mechanism and a position-wise feed-forward network. Using the dot-product self-attention mechansim, for a given layer, the queries, keys, and values are computed as:

$$\mathbf{Q} = \mathbf{X}_{\text{embed}}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}_{\text{embed}}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}_{\text{embed}}\mathbf{W}_V$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learned projection matrices. The self-attention for one head is given by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

with d_k being the dimensionality of the queries/keys. With h heads, the outputs are concatenated and projected to form the final output of the attention module:

$$\text{MultiHead}(\mathbf{X}_{\text{embed}}) = \mathbf{W}_O \,\text{concat}\left(\text{head}_1, \ldots, \text{head}_h\right),$$

where $\mathbf{W}_O$ is the output projection matrix. Each encoder layer also contains a position-wise feed-forward network:

$$\text{FFN}(\mathbf{z}) = \max(0, \mathbf{z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$

This is applied independently to each position. Both the multi-head attention and the feed-forward network include residual connections and layer normalization (post-layer normalization) to stabilize training:

$$\mathbf{z}' = \text{LayerNorm}(\mathbf{X}_{\text{embed}} + \text{MultiHead}(\mathbf{X}_{\text{embed}})),$$

$$\mathbf{z}'' = \text{LayerNorm}(\mathbf{z}' + \text{FFN}(\mathbf{z}'))$$

Therefore, the transformer encoder learns to capture complex dependencies in the data by assigning attention scores that highlight important temporal relationships.

After processing through the transformer encoder, the resulting latent representation is mapped back to the original input space using a linear decoder:

$$\hat{\mathbf{X}} = \mathbf{W}_{\text{out}}\mathbf{Z} + \mathbf{b}_{\text{out}}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d\times d_{\text{model}}}$ is the decoder weight matrix,and $\mathbf{b}_{\text{out}} \in \mathbb{R}^d$ is the decoder bias.

Similarly to our Autoencoder model, the SAITS model is trained to minimize the reconstruction error, again through the use of the Mean Squared Error (MSE) loss. The loss accounts for both time steps and features, hence the normalization by $T \times d$:

$$\mathcal{L}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{T\times d}\sum_{t=1}^{T}\sum_{i=1}^{d}\left(x_{t,i} - \hat{x}_{t,i}\right)^2$$

Although the SAITS model can learn intricate relationships among features for imputation, we were wary of our hardware constraints and the possibility of overfitting. Therefore we included the regularization techniques of dropout

($= 0.1$) and weight decay ($1e^{-5}$. This helped the model generalize better as we previously struggled with the high variance due to the errors from the sensitivity to small fluctuations in the training set. When run over the testing set, the model previously underperformed (RMSE ≈ 0.24) without adjusting for this bias-variance tradeoff.

This provides a complementary baseline to the autoencoder, with a different emphasis on capturing relationships among features.

5. Contrastive Learning & Graph Regularization Model

Our CL×GR model aims to learn latent representations that are invariant to data augmentation while respecting the spatial and temporal proximity of oil tanks. We combine a contrastive loss with a graph-based regularization term to encode domain knowledge about the tanks' proximity, enhancing imputation performance.

We use an encoder $f(\cdot)$, implemented as a multi-layer perceptron (MLP) with two hidden layers (128 neurons each with ReLU activations), to obtain latent embeddings from the input data. A projection head $g(\cdot)$, consisting of two linear layers (128 to 64 units with a ReLU activation in between), maps these embeddings into a lower-dimensional space where the contrastive loss is computed. The use of a separate projection head is inspired by SimCLR (Chen et al., 2020), which demonstrated improved performance in contrastive learning. The maximum use of 128 neurons is to satisfy our hardware constraints, once again.

Given two augmented views $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_i^{(2)}$ of the same sample, their embeddings and projections are computed as:

$$\mathbf{h}_i^{(1)} = f\left(\mathbf{x}_i^{(1)}\right), \quad \mathbf{h}_i^{(2)} = f\left(\mathbf{x}_i^{(2)}\right)$$

$$\mathbf{z}_i^{(1)} = g\left(\mathbf{h}_i^{(1)}\right), \quad \mathbf{z}_i^{(2)} = g\left(\mathbf{h}_i^{(2)}\right)$$

The embeddings are normalized ($\mathbf{z}_i = \mathbf{z}_i/\|\mathbf{z}_i\|_2$, $\mathbf{h}_i = \mathbf{h}_i/\|\mathbf{h}_i\|_2$) before computing losses. The similarity between projections is measured by the **cosine similarity**:

$$\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\|_2 \|\mathbf{z}_j\|_2}$$

which simplifies to $\mathbf{z}_i^\top \mathbf{z}_j$ because $\mathbf{z}_i$ $\mathbf{z}_j$ are normalized ($\|\mathbf{z}_i\|_2 = 1$ and $\|\mathbf{z}_j\|_2 = 1$). Normalizing the embeddings simplifies cosine similarity to a dot product, reducing overhead for our contrastive loss function. The contrastive loss for a batch $\mathcal{B}$ is defined as:

$$L_{\text{contrastive}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} -\log \frac{\exp\left(\text{sim}\left(\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)}\right)/\tau\right)}{\sum_{j \in \mathcal{B}} \exp\left(\text{sim}\left(\mathbf{z}_i^{(1)}, \mathbf{z}_j^{(2)}\right)/\tau\right)},$$

where $\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i \cdot \mathbf{z}_j$, and τ is the temperature parameter. A lower τ sharpens the softmax distribution, focusing on the most similar positive pairs, while a higher τ smooths the distribution.

5.1. Graph Regularization

We construct a graph over the tanks based on spatial and temporal proximity. For spatial edges, we compute the Haversine distance between tanks using their latitude and longitude coordinates (extracted from the *Location* column) and create an edge between two tanks if their distance is less than a threshold $d_{\text{threshold}}$, which is tuned during cross-validation. The spatial adjacency matrix A is defined such that $A_{ij} = 1$ if the distance between tanks i and j is less than $d_{\text{threshold}}$, and $A_{ij} = 0$ otherwise.

For temporal edges, we connect consecutive time steps for each tank based on the `imaging_time` column, forming an undirected graph where each tank's observations are linked in temporal order. Specifically, for each tank, we sort its observations by time and create an edge between indices t and $t+1$, making the connection bidirectional (i.e., $(t, t+1)$ and $(t+1, t)$).

The graph penalty encourages embeddings of connected tanks to be similar, with separate penalties for spatial and temporal edges:

$$L_{\text{graph}} = \lambda \cdot \frac{1}{|E_{\text{spatial}}| + |E_{\text{temporal}}|} \left(L_{\text{spatial}} + L_{\text{temporal}}\right),$$

where:

$$L_{\text{spatial}} = \sum_{(i,j) \in E_{\text{spatial}}} w_{ij} \|\tilde{f}(\mathbf{x}_i) - \tilde{f}(\mathbf{x}_j)\|_2^2,$$

$$L_{\text{temporal}} = \sum_{(i,j) \in E_{\text{temporal}}} \|\tilde{f}(\mathbf{x}_i) - \tilde{f}(\mathbf{x}_j)\|_2^2,$$

and $\tilde{f}(\mathbf{x}_i) = f(\mathbf{x}_i)/\|f(\mathbf{x}_i)\|_2$ are the normalized embeddings, E_{spatial} and E_{temporal} are the sets of spatial and temporal edges within the batch, and λ is the regularization hyperparameter. The spatial penalty is weighted by the average degree centrality of the two tanks, defined as:

$$w_{ij} = \frac{\text{centrality}_i + \text{centrality}_j}{2}$$

where centrality_i is the normalized degree of tank i in the spatial adjacency graph—the number of neighbors of tank i divided by the maximum degree in the graph—and 10^{-6} prevents division by zero. This weighting was used because initial tests did not ensure that highly connected tanks had a stronger influence on the regularization. Tanks with many neighbors are likely to be in dense clusters, where spatial relationships are more significant. Weighting their contribution more heavily ensures that the model prioritizes these clusters.

We initially chose degree centrality for its simplicity and ability to emphasize tanks in dense spatial clusters like tank farms, and explored eigenvector centrality as an alternative to capture the influence of tanks connected to other highly connected tanks, better reflecting the hierarchical structure of spatial relationships.

Eigenvector centrality measures a tank's importance based on the principle that connections to highly connected tanks contribute more to its score than connections to less connected ones. Formally, for a tank i, its eigenvector centrality is the i-th component of the principal eigenvector of the spatial adjacency matrix A, satisfying $A\mathbf{v} = \lambda_{\max}\mathbf{v}$, where $\lambda_{\max}$ is the largest eigenvalue. We normalize the centrality scores by dividing by the maximum score to ensure they lie in $[0, 1]$. The weight w_{ij} for a spatial edge (i, j) is then computed as:

$$w_{ij} = \frac{\text{eigenvector\_centrality}_i + \text{eigenvector\_centrality}_j}{2}$$

where $\text{eigenvector\_centrality}_i$ is the normalized eigenvector centrality of tank i. This approach emphasizes the influence of tanks that are part of densely connected clusters, potentially capturing more nuanced spatial relationships than degree centrality. We compare the performance of both centrality measures during hyperparameter tuning, as discussed in Section 6.2, to determine which enhances performance. Our total loss becomes:

$$L = L_{\text{contrastive}} + L_{\text{graph}}$$

5.2. Augmentation and Training

For data augmentation, we first preprocess the dataset by filling missing values with zeros and normalizing the features. During training, for each sample, we generate two augmented views by masking 20% of the features in accordance with the probabilities from our analysis in Section 2.2 and adding Gaussian noise with a standard deviation of 0.01. This augmentation is applied uniformly across all features, regardless of whether they were originally missing, enabling the model to learn robust representations.

After training, we use the learned encoder $f(\cdot)$ to compute embeddings for all samples in the dataset. To impute missing `fill_pct` values, we apply K-nearest neighbors (KNN) regression in the embedding space. Specifically, we train a KNN regressor on the embeddings of samples with known `fill_pct` values and use it to predict the missing `fill_pct` values for the remaining samples. Using KNN in the latent space is distinct from applying KNN directly to the raw features because it now leverages the similarity between tanks that are geographically or temporally close, as encoded in the embeddings, rather than relying solely on feature similarity in the original space.

We tune the hyperparameters, including the regularization weight λ, temperature τ, distance threshold $d_{\text{threshold}}$, and learning rate, using walk-forward cross-validation on the training set to optimize imputation performance. For clarity, below is the pseudocode of our algorithm used for training:

Algorithm 1 Contrastive Learning & Graph Regularization

Input: Dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, graph $G = (V, E)$, temperature τ, regularization weight λ, number of epochs T
for epoch $= 1$ to T **do**
  for each mini-batch $\mathcal{B} \subset \mathcal{X}$ **do**
    Augment: For each $\mathbf{x}_i \in \mathcal{B}$, generate two views:

$$\mathbf{x}_i^{(1)} \leftarrow \text{augment}(\mathbf{x}_i), \quad \mathbf{x}_i^{(2)} \leftarrow \text{augment}(\mathbf{x}_i)$$

    Encode: Compute embeddings:

$$\mathbf{h}_i^{(1)} = f\big(\mathbf{x}_i^{(1)}\big), \quad \mathbf{h}_i^{(2)} = f\big(\mathbf{x}_i^{(2)}\big)$$

    Project: Compute projections:

$$\mathbf{z}_i^{(1)} = g\big(\mathbf{h}_i^{(1)}\big), \quad \mathbf{z}_i^{(2)} = g\big(\mathbf{h}_i^{(2)}\big)$$

    Normalize: Normalize embeddings:

$$\mathbf{z}_i^{(1)} \leftarrow \mathbf{z}_i^{(1)}/\|\mathbf{z}_i^{(1)}\|_2, \quad \mathbf{z}_i^{(2)} \leftarrow \mathbf{z}_i^{(2)}/\|\mathbf{z}_i^{(2)}\|_2,$$

$$\mathbf{h}_i^{(1)} \leftarrow \mathbf{h}_i^{(1)}/\|\mathbf{h}_i^{(1)}\|_2, \quad \mathbf{h}_i^{(2)} \leftarrow \mathbf{h}_i^{(2)}/\|\mathbf{h}_i^{(2)}\|_2.$$

    Contrastive Loss: Compute:

$$L_{\text{contrastive}} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} -\log\frac{\exp\big(\text{sim}(\mathbf{z}_i^{(1)}, \mathbf{z}_i^{(2)})/\tau\big)}{\sum_{j\in\mathcal{B}}\exp\big(\text{sim}(\mathbf{z}_i^{(1)}, \mathbf{z}_j^{(2)})/\tau\big)}.$$

    Graph Loss: For all pairs $(i, j) \in E_{\text{spatial}} \cap \mathcal{B}^2$, compute:

$$L_{\text{spatial}} = \sum_{(i,j)\in E_{\text{spatial}}\cap\mathcal{B}^2} w_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|^2,$$

    For all pairs $(i, j) \in E_{\text{temporal}} \cap \mathcal{B}^2$, compute:

$$L_{\text{temporal}} = \sum_{(i,j)\in E_{\text{temporal}}\cap\mathcal{B}^2} \|\mathbf{h}_i - \mathbf{h}_j\|^2,$$

$$L_{\text{graph}} = \lambda\cdot\frac{1}{|E_{\text{spatial}}\cap\mathcal{B}^2| + |E_{\text{temporal}}\cap\mathcal{B}^2|}\left(L_{\text{spatial}} + L_{\text{temporal}}\right).$$

    Total Loss: Set:

$$L = L_{\text{contrastive}} + L_{\text{graph}}.$$

    Backpropagate and update model parameters.
  end for
end for
Output: Trained encoder $f(\cdot)$.

where $\mathcal{B}^2$ denotes all possible pairs of samples in the batch. Although graph edges E are defined over the entire dataset,

we only have access to a subset of samples during mini-batch training. Therefore, we filter the edges to include only those where both endpoints are in $\mathcal{B}^2$, i.e. $E \cap \mathcal{B}^2$.

6. Results

6.1. Evaluation Metrics & Cross Validation

Imputation quality is judged with the same metrics used in (Khan, 2024). *RMSE* reflects the overall magnitude of errors and penalizes large deviations that can propagate into subsequent analyses; *MAE* offers a scale–preserving measure that is robust to a few extreme gaps in our dataset; *MAPE* expresses the error as a percentage of the true value, making performances directly comparable across variables; and R^2 indicates how much of the missing–value variance the imputer is able to reconstruct.

As an imputer that is accurate on average may still introduce systematic over- or underestimation, we also test for bias. Let $d_i = \hat{x}_i - x_i$ be the difference between imputed and observed values in the artificially masked positions. The null hypothesis of no bias, $H_0 : \mu_d = 0$, is evaluated with the paired t-statistic

$$t = \frac{\bar{d}}{s_d/\sqrt{N}}, \qquad \bar{d} = \tfrac{1}{N}\sum_{i=1}^{N} d_i, \quad s_d^2 = \tfrac{1}{N-1}\sum_{i=1}^{N}(d_i - \bar{d})^2,$$

which follows a t-distribution with $N-1$ degrees of freedom. A p-value larger than 0.05 indicates that the imputer does not shift the distribution of *fill_pct*.

To gauge whether the latent space learned by CL×GR is useful for downstream tasks, we evaluate two neighbourhood-based criteria. Firstly, **Trustworthiness**(Venna and Kaski, 2006) measures how faithfully the k nearest neighbors in the high-dimensional space are preserved after projection. More specifically:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in U_k(i)} \big(r(i,j) - k\big),$$

where $U_k(i)$ is the set of points that are among the k nearest neighbors of i in 2-D but not in the original space, and $r(i,j)$ is the rank of point j among all points with respect to point i in the original space. For clarity, $T(k) \in [0,1]$. We also incorporate **k-NN Tank-ID purity** which quantifies cluster homogeneity:[1]

$$P(k) = \frac{1}{nk} \sum_{i=1}^{n} \sum_{j \in N_k(i)} \mathbf{1}\big[c_i = c_j\big],$$

where $N_k(i)$ are the k nearest neighbours of point i in 2-D and c_i is its tank label. $P = 1$ indicates perfectly separated tanks; random mixing yields $P \approx 1/\text{number of tanks}$.

[1] A lightweight alternative to the more expensive silhouette, which our machine struggled with on 36k points.

6.2. Performance Comparison

All models, including the simple imputation techniques discussed in Section 4.1, apply the same stratified masking to the known 48.85% of `fill_pct` values. The target masking rate was 20%, adjusted for the natural missingness rate, resulting in an effective masking rate of approximately 18.7–19.0% due to the stratified probabilities based on `season` and `tank_id` (Section 2.2).

With a 70/30 train-test split, which preserves temporal order, this results in a training set of 25,225 rows with 12,400 non-missing `fill_pct` values, and a test set of 10,811 rows with 5,202 non-missing `fill_pct` values. In the test set, this masking corresponds to 986 artificially masked values, and in the full dataset, 3227 values. We preprocessed the dataset by normalizing the input features using the mean and standard deviation of the training set and filling missing values with zeros to ensure consistent input to the model.

All three models were placed through rigorous 5 fold walk-forward grid-search cross-validation. For the Autoencoder, the hyperparameters tuned were the learning rate and encoding dimension. For SAITS, the hyperparameters included the model dimension, learning rate, number of attention heads, and number of transformer layers. For the CL×GR, the hyperparameters included the graph regularization weight (λ_{graph}), temperature for contrastive loss, distance threshold for graph construction, learning rate, centrality type (eigenvector/degree), and number of neighbors. See Appendix A for detailed cross-validation metrics.

Following cross-validation, the selected hyperparameters were used to train each model on the entire training set and evaluate performance on the test set, before being applied to the full dataset to impute all missing values.

At 51.15% observed data none of the individual covariates correlate strongly with `fill_pct` ($|\rho| \leq 0.20$, MI ≤ 0.15). Despite this difficulty, our CL×GR model achieves the lowest error across all metrics in all sets. For the full dataset, which we highlight as our goal is to correct the missingness across the entirety of the `Louisiana Tank Farm` dataset, the RMSE (0.211) decreased 0.5% against AE and 1.8% against SAITS. MAE (0.171) decreased 4.6% compared to AE and 5.8% to SAITS. Most significantly, MAPE (42%) decreased over 11.57% against both baseline models. Additionally, it retained unbiased estimates ($p = 0.56$).

The modest R^2 (0.11) is understandable: the series itself has high intrinsic variance, with $\sigma \approx 0.22$, and the imputation task is extremely sparse. Nevertheless, the R^2 improved considerably compared to the SAITS and autoencoder models (Table 3).

	Train					Test					Full dataset				
Method	**RMSE**	**MAE**	**MAPE (%)**	**R^2**	**Bias p-value**	**RMSE**	**MAE**	**MAPE (%)**	**R^2**	**Bias p-value**	**RMSE**	**MAE**	**MAPE (%)**	**R^2**	**Bias p-value**
Autoencoder	0.225	0.190	51.396	0.001	0.745	0.215	0.182	46.735	-0.001	0.687	0.212	0.179	46.867	0.085	0.873
SAITS	0.222	0.187	49.092	-0.004	0.990	0.218	0.185	46.223	0.001	0.114	0.215	0.181	47.027	0.029	0.116
CLxGR	0.204	0.166	42.274	0.150	0.697	0.201	0.171	42.097	0.133	0.894	0.211	0.171	42.007	0.112	0.565

Table 3: Comparison of imputation methods across train, test, and final evaluations.

	CL×GR							CL						
Metric	**RMSE**	**MAE**	**MAPE (%)**	**R^2**	**Bias p-value**	**Trust.**	**Purity**	**RMSE**	**MAE**	**MAPE (%)**	**R^2**	**Bias p-value**	**Trust.**	**Purity**
Value	0.211	0.171	42.008	0.112	0.565	0.998	0.010	0.212	0.171	42.015	0.112	0.570	0.981	0.010

Table 4: Comparison of CL×GR and CL on Final Set. Trust. = Trustworthiness, Purity = Tank-ID Purity.

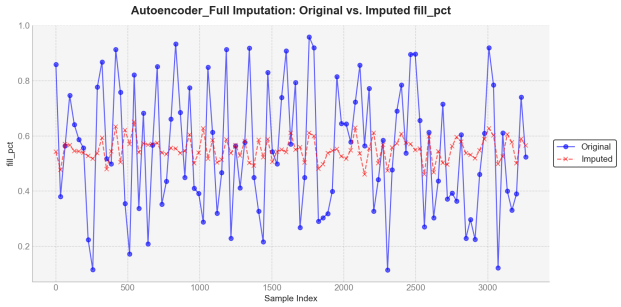

Figure 15: Sample of imputed values compared to masked values using the autoencoder model.

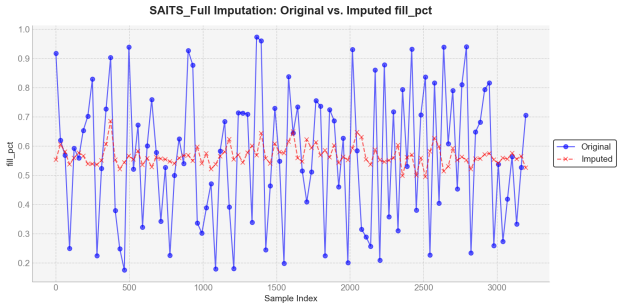

Figure 16: Sample of imputed values compared to masked values using the SAITS model.

One can see the improved performance of the CL×GR when comparing Figure 17 to Figure 15 and Figure 16. Over the selected sample, the CL×GR aligns much closer with the variability in the underlying `fill_pct` time series, explaining its significant R^2 improvement over the SAITS and autoencoder models.

We also conducted an ablation study to compare the contrastive learning agent's performance with graph regularization ($\lambda = 1000$) and without ($\lambda = 0$). While the discrepancy in RMSE was small (0.211 vs 0.212), the embedding quality, represented by trustworthiness, increased from 0.981 to 0.998, suggesting benefits for any down-

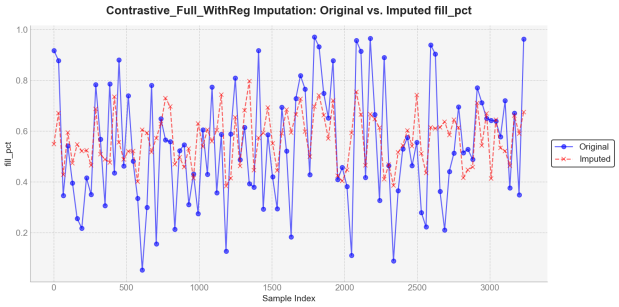

Figure 17: Sample of imputed values compared to masked values using the Contrastive Loss model with Graph Regularization.

stream tasks that reuse the latent space.

Therefore, the graph term mainly improves *representation quality*——beneficial for any task that reuses the embedding——while leaving point-wise error almost unchanged. The improved structure is noticeable in Figure 18, where we use t-SNE to construct the plot. The t-SNE method constructs a 2-D visualization by first modeling pairwise similarities in high-dimensional space using Gaussian distributions, then minimizing the Kullback-Leibler Divergence between these and a t-distribution in 2-D to preserve local neighborhood structure. Our second metric, Tank-ID purity (≈ 0.01), remains low in both settings, indicating that individual tanks overlap in latent space.

The marginal performance discrepancies in Figure 4 are unsurprising given the weak per-feature correlations (≈ 0.1 Spearman correlation or mutual information) and the aggressive 51.15% missingness. This finding aligns with prior work in graph-augmented contrastive learning, where the regularization benefits are contingent on the graph capturing task-relevant relationships (Zhu et al., 2020).

Spearman correlation is computed on pairwise complete observations, meaning rows where both `fill_pct` and the feature i are non-missing. With 51.15% missingness

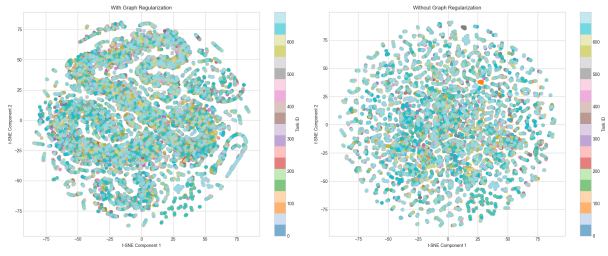

Figure 18: Latent space comparison when graph regularization is applied (Left) versus not (right).

in `fill_pct`, we're working with 48.85% of the data for these calculations, reducing the sample size and potentially the statistical power to detect stronger correlations. However, assuming the remaining data was representative, as we did, the Spearman correlation, being non-parametric and robust to missingness, should have provided accurate representations.

We also compared the hardware performance of our 3 advanced methods, discovering the CL×GR provided a strong mid-way performance between the autoencoder and SAITS models in terms of inference time and memory usage. See Appendix B for detailed cross-validation metrics.

7. Discussion

Our work contributes to existing imputation research in several ways. Firstly, we addressed our research question (Section 1.2) by demonstrating that graph regularization provides a slight improvement over contrastive learning alone when features with ≈ 0.1 Spearman correlation or mutual information are used, as seen in the slight reduction in RMSE on the Louisiana Tank Farm dataset. Second, our CL×GR outperformed other state-of-the-art models, including the SAITS and autoencoder, across RMSE, MAE, MAPE, and R^2. Thirdly, we achieved a $\sim$10% reduction in absolute error on an extremely sparse, weak-signal panel, and produced a more faithful latent geometry for down-stream analysis of the `Louisiana Tank Farm` dataset (Figure 18). This provides evidence that graph-aware contrastive learning is a worthwhile addition to the imputation toolkit.

7.1. Limitations and Assumptions

Our study has several limitations and relies on key assumptions that warrant discussion. Firstly, the primary limitation of our CL×GR approach is the weak correlation between `fill_pct` and the spatial-temporal features used to construct the graph. Although among the strongest monotonic (Section 3.2] & 2.3) and non-linear (Section 3.6 relationships with `fill_pct`, spatial proximity and temporal consecutiveness were not strong predictors of tank fill levels. This explains the minor differences in our abalation study when $\lambda_{graph} = 0$ and $\lambda_{graph} = 100$.

This finding aligns with prior work, such as GRACE (Zhu et al., 2020), which notes that graph regularization benefits are contingent on the graph capturing task-relevant relationships. For instance, `distance_to_nearest_tank_id` showed a Spearman correlation of 0.122 and MI of 0.104. While one of our strongest features, these values also indicate that spatial proximity may not be a solid predictor of tank fill levels. In our case, these weak dependencies likely limited the graph regularization's ability to improve imputation performance, though it did marginally enhance latent space quality.

The 51.15% missingness in `fill_pct` posed a significant challenge for our correlation and MI analyses, which we conducted before imputation to avoid introducing bias and given the intent of our research. Even simple imputation methods like KNN assume row similarity and mean imputation assumes MCAR, both of which would have inflated correlations and mislead our feature selection. By analyzing pre-imputation data, we preserved the true relationships but worked with only 48.85% of the data for pairwise complete observations. This reduced sample size likely decreased the statistical power to detect stronger correlations.

Additionally, our missingness pattern is MAR (Section 2.2), meaning the probability of missingness depends on *observed* variables but not on the missing values themselves. While the Spearman correlation is robust to missing data, MAR can still bias relationships—for instance, if `fill_pct` is more likely to be missing during winter (when `temperature_min` is low), the true correlation between these variables may be underestimated. To test this, future work could impute `fill_pct` using our contrastive learning (CL) model and recompute correlations, assessing whether missingness attenuated the observed effects.

Our dataset violates the normality and independence assumptions of standard correlation and MI methods due to non-normal distributions and temporal autocorrelation within tanks, as well as spatial correlation across regions. We addressed non-normality by using Shapiro-Wilk tests and Q-Q plots to confirm deviations, and we adopted Spearman correlation, a non-parametric method, to handle this violation. For autocorrelation, we used TimeSeriesSplit to preserve chronological order and added lagged and rolling features to embed temporal dynamics, mitigating i.i.d. violations. While these adjustments were robust, they assume that the rank-based Spearman correlation and k-NN-based MI estimator can fully capture the relevant dependencies,

which may not hold for complex non-linear relationships not addressed by our feature engineering.

Furthermore, our feature set included lagged and rolling features to capture temporal dynamics, but we did not explore interaction terms, such as temperature_min $*$ wti_price, which might have revealed stronger relationships with `fill_pct`.

7.2. Future Work

To address the limitations identified, several directions for future research are proposed. Firstly, alternative imputation methods in the latent space could be explored. Our current approach uses KNN regression on the embeddings, but a neural network-based imputation method might better capture complex patterns in the latent space, potentially improving RMSE. Second, incorporating a graph neural network (GNN) could enhance the model's ability to leverage graph structure, though this would require balancing the computational advantages of our current MLP-based model, which is more lightweight.

Third, revisiting feature engineering to include interaction terms or domain-specific features, such as tank capacity or regional demand, could uncover stronger relationships with `fill_pct`, potentially improving both CL and GR performance. Fourth, alternative graph structures based on operational similarities or economic factors could better capture `fill_pct`'s drivers, potentially enhancing CL×GR's performance. Finally, exploring alternative centrality measures beyond degree and eigenvector centrality, such as betweenness or closeness centrality, may improve the graph penalty's effectiveness.

References

Amirteimoori, A. and Kordrostami, S. (2010). A Euclidean distance-based measure of efficiency in data envelopment analysis. *Optimization*, 59(7):985–996.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *Proceedings of the 37th International Conference on Machine Learning*, 119:1597–1607. [Submitted: 13 Feb 2020 (v1), Revised: 1 Jul 2020 (v3)].

Du, W., Cote, D., and Liu, Y. (2023). SAITS: Self-attention-based imputation for time series. *Expert Systems with Applications*, 219:119619.

Fang, L., Xiang, W., Zhou, Y., Fang, J., Chi, L., and Ge, Z. (2023). Dual-branch cross-dimensional self-attention-based imputation model for multivariate time series. *Knowledge-Based Systems*, 279:110896.

Gondara, L. and Wang, K. (2018). MIDA: Multiple Imputation Using Denoising Autoencoders. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018*, pages 260–272. Springer.

Good, P. (2005). *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer Series in Statistics. Springer, New York, 2nd edition. Contains valuable techniques for reducing computation time, practical advice on experimental design, comparisons with bootstrap, parametric, and nonparametric techniques; Includes a three-part bibliography featuring more than 1,000 articles.

Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual Review of Psychology*, 60:549–576.

Hyndman, R. J. and Athanasopoulos, G. (2021). *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 3rd edition. Open-access textbook.

Jiao, X., Li, Y., Zhai, Y., and Yang, Y. (2023). Improving Generalization for Missing Data Imputation via Dual Corruption Denoising Autoencoders. *arXiv preprint arXiv:2309.11384*.

Khan, M. A. (2024). A comparative study on imputation techniques: Introducing a transformer model for robust and efficient handling of missing EEG amplitude data. *Bioengineering*, 11(8). (This article belongs to the Special Issue Intelligent IoMT Systems for Brain–Computer Interface).

Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Physical Review E*, 69:066138. 16 pages, including 18 figures; Submitted on 28 May 2003; Also appears in: Phys. Rev. E 69, 066138 (2004).

Lee, S., Lee, S., Lee, J., Lee, W., and Son, Y. (2024). Graph contrastive learning with consistency regularization. *Pattern Recognition Letters*, 181:43–49.

Ndifon, V. B. (2023). *The Reliability and Efficiency of Replacing Missing Data in Sparse Data Sets*. Phd thesis, Northcentral University, Scottsdale, AZ, USA. [Google Scholar].

Nijman, S. W., Hoogland, J., Groenhof, T. K. J., Brandjes, M., Jacobs, J. J., Bots, M. L., Asselbergs, F. W., Moons, K. G., and Debray, T. P. (2021). Real-time imputation of missing predictor values in clinical practice. *European Heart Journal - Digital Health*, 2(1):154–164. [Google Scholar] [CrossRef].

Pan, L., Ren, Q., Li, Z., and Lv, X. (2025). Rethinking Spatial-Temporal Contrastive Learning for Urban Traffic Flow Forecasting: Multi-Level Augmentation Framework. *Complex & Intelligent Systems*, 11:96.

Razali, N. M. and Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33.

Schober, P., Boer, C., and Schwarte, L. A. (2018). Correlation coefficients: Appropriate use and interpretation. *Anesthesia & Analgesia*, 126(5):1763–1768. Author Information: Patrick Schober, MD, PhD, MMedStat; Christa Boer, PhD, MSc; Lothar A. Schwarte, MD, PhD, MBA.

Schwerter, J., Gurtskaia, K., Romero, A., Zeyer-Gliozzo, B., and Pauly, M. (2024). Evaluating tree-based imputation methods as an alternative to MICE PMM for drawing inference in empirical studies. *arXiv preprint arXiv:2401.09602*.

Shi, Y., Wan, J., Zhang, X., and Yin, Y. (2023). CL-Impute: A contrastive learning-based imputation for dropout single-cell RNA-seq data. *Computers in Biology and Medicine*, 164:107263.

van Buuren, S. (2018). *Flexible Imputation of Missing Data*. Chapman & Hall/CRC Interdisciplinary Statistics. Chapman and Hall/CRC, New York, 2nd edition. eBook published July 16, 2018.

Venna, J. and Kaski, S. (2006). Local multidimensional scaling. *Neural Networks*, 19(6–7):889–899. Special Issue.

Wang, D., Yan, Y., Qiu, R., Zhu, Y., Guan, K., Margenot, A. J., and Tong, H. (2023). Networked time series imputation via position-aware graph enhanced variational autoencoders. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*.

White, I. R. and Carlin, J. B. (2010). Bias and efficiency of multiple imputation compared with complete-case analysis for missing covariate values. *Statistics in Medicine*, 29(28):2920–2931.

Wi, H., Shin, Y., and Park, N. (2024). Continuous-time autoencoders for regular and irregular time series imputation. In *Proceedings of the ACM Conference*, New York, NY, USA. Association for Computing Machinery. Copyright held by the owner/author(s). Publication rights licensed to ACM.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. (2020). Deep graph contrastive representation learning. *[Journal/Conference Name]*. * Equal contribution, † Corresponding author.

Table 5: Cross-Validation Results for Autoencoder, SAITS, and CLxGR Models (Top 5 Configurations by RMSE)

Autoencoder					SAITS					CLxGR				
Hyperparameters	RMSE	MAE	MAPE	R^2	Hyperparameters	RMSE	MAE	MAPE	R^2	Hyperparameters	RMSE	MAE	MAPE	R^2
$d = 512, lr = 0.001$	0.01	0.01	2.10	0.99	$d_m = 64, h = 8, l = 4, lr = 0.001$	0.044	0.029	7.68	0.96	$(\lambda_g = 100, t = 0.05, d_t = 20, lr = 0.001, c = d, k = 7)$	0.20	0.16	38.43	0.19
$d = 256, lr = 0.001$	0.02	0.015	3.56	0.99	$d_m = 32, h = 4, l = 4, lr = 0.001$	0.076	0.057	14.70	0.88	$(\lambda_g = 1000, t = 0.05, d_t = 20, lr = 0.001, c = e, k = 7)$	0.20	0.16	40.04	0.22
$d = 128, lr = 0.001$	0.04	0.03	7.08	0.96	$d_m = 16, h = 4, l = 2, lr = 0.001$	0.09	0.07	18.54	0.82	$(\lambda_g = 100, t = 0.05, d_t = 72.88, lr = 0.001, c = e, k = 7)$	0.1995	0.1597	37.69	0.174
$d = 64, lr = 0.001$	0.0840	0.0658	16.62	0.854	$d_m = 8, h = 4, l = 4, lr = 0.001$	0.17	0.14	36.64	0.38	$(\lambda_g = 1000, t = 0.05, d_t = 72.88, lr = 0.001, c = e, k = 7)$	0.2085	0.1696	43.07	0.160
$d = 32, lr = 0.001$	0.1414	0.1167	30.43	0.588	$d_m = 8, h = 4, l = 2, lr = 0.001$	0.17	0.14	37.92	0.37	$(\lambda_g = 1000, t = 0.05, d_t = 20, lr = 0.001, c = d, k = 7)$	0.21	0.17	42.90	0.15

d: encoding_dim; d_m: d_model, h: nhead, l: num_layers; λ_g: lambda_graph, t: temperature, d_t: distance_threshold, lr: learning rate, c: centrality_type (d=degree, e=eigenvector), k: n_neighbors.
RMSE, MAE, MAPE, and R^2 are averages across 5-fold cross-validation. Given CV folds here results appear stronger for autoencoder, but run over the full training set they appeared as in Table 3. SAITS model struggled given hardware on larger d_m, hence we only included the top two that were measurable.

8. Appendix

A. Cross-Validation Results

B. Hardware Performance

Table 6: Hardware Performance Metrics

Metric	Autoencoder	SAITS	CLxGR
Hyperparams	d=512	d_m=64, h=8, l=4	λ_g=0, t=0.05, d_t=20 lr=0.001, c=d, k=7
Train Time (s)	7.51	412.19	107.13
Train Mem (MB)	155.97	241.06	180.38

d: encoding_dim; d_m: d_model; h: nhead; l: layers; λ_g: graph weight; t: temperature; d_t: distance threshold; lr: learning rate; c: centrality type (d=degree); k: n_neighbors.