Vol. 2, No.2 , pp.69-84 , 2024

Open Industrial Sensor Graph Dataset for Evaluating Time-Series Data Engineering Pipelines

Daniel R. Okafor[1], Yiwen Tang[2, *], Sofia Mäkinen[1], Rahul Banerjee[3]

[1] Institute for Data Systems and Learning, Norwich NR4 7TJ, United Kingdom

[2] Department of Computing, Imperial Data Science Lab, London SW7 2AZ, United Kingdom

[3] School of Computer Science and Engineering, Bengaluru 560012, India

* *yiwen.tang@idsl.ac.uk*

Article Information	
Received	14 January 2024
Accepted	28 May 2024
DOI	https://doi.org/10.63646/datamind.2024.020206

Abstract

Industrial Internet-of-Things deployments now instrument plants with thousands of networked sensors whose readings arrive at irregular rates, with gaps, and with strong cross-sensor dependence induced by physical topology. For analytics on such data, the dominant bottleneck is increasingly not modelling accuracy but the data-engineering pipeline that ingests, validates, aligns, imputes, stores, indexes, and serves the multivariate stream. Existing open time-series benchmarks were built to evaluate forecasting or classification models on clean, regularly sampled, and essentially flat collections of series; they neither preserve the sensor graph nor exercise the pipeline end-to-end, and they rarely ship the schema, data dictionary, and persistent identifiers needed for reproducible systems research. We introduce ISGraph, an open Industrial Sensor Graph dataset and benchmark designed specifically to evaluate time-series data-engineering pipelines. ISGraph couples a physically grounded generative model of plant sensor networks with explicit node and edge tables, controlled fault injection, and controlled missingness and irregular sampling. It is distributed with a normalised relational-plus-graph schema, a complete field-level data dictionary, a reference pipeline implementation, a query API, and a permanent repository deposit. We define a benchmark protocol over five pipeline tasks—schema validation, temporal alignment, graph-aware imputation, feature extraction, and windowed query serving—with control baselines, throughput and latency metrics, and reconstruction-error analysis. Using the reference pipeline we show that graph-aware imputation lowers reconstruction RMSE by 21–39% relative to forward-fill and to competitive flat baselines, that the advantage grows with node degree and with gap length, and that a columnar store augmented with a graph-neighbourhood index sustains the highest ingestion throughput while delivering the lowest windowed-query latency. ISGraph is intended as shared infrastructure for database, data-engineering, and computational-discovery research on industrial time series.

Keywords: ***Industrial IoT; sensor graph; time-series data engineering; data pipelines; database schema;***

missing-data imputation; benchmark dataset; reproducibility

1. Introduction

Modern industrial facilities are dense measurement systems. A mid-sized process plant, a wind farm, or an aero-engine test cell carries hundreds to thousands of sensors that report temperatures, pressures, flows, vibrations, and electrical quantities. These sensors are not statistically independent channels that happen to be recorded side by side. They are wired into a physical and functional structure: a pressure transducer downstream of a pump responds to that pump with a characteristic lag, two thermocouples on the same heat exchanger move together, and a fault in one rotating component propagates through the mechanically coupled stages around it (Akyildiz et al., 2002; Lei et al., 2018). The natural mathematical object for such a deployment is therefore not a matrix of unrelated columns but a graph whose nodes are sensors and whose edges encode proximity, shared physics, and causal coupling (Shuman et al., 2013; Ortega et al., 2018).

The practical difficulty in working with this data is rarely the final predictive model. It is everything that has to happen before a model can be trained or a dashboard can be drawn. Industrial streams arrive at heterogeneous and drifting sampling rates, contain gaps caused by network loss and maintenance, carry sensor-specific noise and saturation, and must be aligned onto a common temporal grid before any cross-sensor computation is meaningful. This sequence of operations—ingestion, schema validation, temporal alignment and resampling, missing-value imputation, feature extraction, storage, indexing, and query serving—is the data-engineering pipeline, and it is where most engineering effort, most latency, and most silent error actually live (Polyzotis et al., 2018; Sambasivan et al., 2021). When the pipeline mistreats the data, downstream accuracy degrades in ways that no amount of model tuning can recover.

Despite this, the open datasets that the community uses to evaluate methods were overwhelmingly built to test models rather than pipelines. Classification and forecasting archives such as the UCR/UEA collection and the algorithm comparisons built on it provide clean, regularly sampled, and largely independent series that are ideal for ranking classifiers but say nothing about ingestion throughput, alignment error, or query latency (Bagnall et al., 2017; Dau et al., 2019). Industrially realistic datasets do exist—the C-MAPSS and N-CMAPSS turbofan degradation simulations, the Secure Water Treatment testbed, and spacecraft telemetry corpora are widely used—but they were assembled around prognostics or anomaly-detection tasks, they seldom expose the sensor graph explicitly, and they do not ship the schema, data dictionary, and benchmark harness needed to measure a pipeline end-to-end (Saxena et al., 2008; Goh et al., 2017; Hundman et al., 2018; Arias Chao et al., 2021). On the database side, decades of work on time-series management systems, columnar storage, and stream processing have produced the very components a pipeline is built from, yet there is no shared, graph-aware benchmark dataset against which competing pipeline designs can be compared on equal terms (Stonebraker & Çetintemel, 2005; Abadi et al., 2008; Jensen et al., 2017).

This article addresses that gap. We present **ISGraph**, an open Industrial Sensor Graph dataset and benchmark whose explicit purpose is the evaluation of time-series data-engineering pipelines rather than the evaluation of a single model. The engineering problem we target is concrete: given multivariate, irregularly sampled, partially missing readings from a known sensor graph, how should a pipeline ingest, validate, align, impute, store, and serve that data so as to maximise reconstruction quality and query performance under controlled, reproducible conditions? ISGraph makes this question measurable. Its contributions are fourfold. First, it provides a physically grounded generative model that emits multivariate sensor time series together with the ground-truth graph that produced them, with controllable missingness, irregular sampling, and injected faults. Second, it ships a normalised relational-plus-graph schema and a complete field-level data dictionary, so that the data is self-

ISSN: 3071-5601 |

describing and FAIR-compliant (Wilkinson et al., 2016). Third, it defines a benchmark protocol over five pipeline tasks with explicit control baselines, performance metrics, and reconstruction-error analysis. Fourth, it is released with a reference pipeline implementation, a query API, container images, and a permanent repository deposit, so that results are reproducible and the artifact is usable as shared infrastructure.

The remainder of the paper is organised as follows. Section 2 reviews related data resources and positions ISGraph against them. Section 3 states the design goals, the generative model, the pipeline reference architecture, and the database schema. Section 4 presents the data dictionary. Section 5 describes the dataset contents and descriptive statistics. Section 6 defines the benchmark protocol. Section 7 reports reference results and an error analysis using the reference pipeline. Sections 8 and 9 discuss implications and limitations, Section 10 documents data availability and reproducibility, and Section 11 concludes.

2. Background and Related Data Resources

Three families of data resources are adjacent to ISGraph, and each illuminates what a pipeline-oriented benchmark must add. The first family is the model-evaluation archive for univariate and multivariate time series. The UCR/UEA classification archive and the systematic comparisons built on it standardised how classifiers are ranked and exposed how easily apparent progress can be mis-attributed when evaluation is careless (Bagnall et al., 2017; Dau et al., 2019). Parallel work on time-series clustering and on representation and similarity measures provides the algorithmic vocabulary that any feature-extraction stage reuses (Esling & Agon, 2012; Paparrizos & Gravano, 2015), while recurrent architectures such as the long short-term memory network underpin many sequence models that consume pipeline output (Hochreiter & Schmidhuber, 1997). A recurring lesson from this literature is methodological rather than algorithmic: benchmarks shape the field, and flawed benchmarks create the illusion of progress (Wu & Keogh, 2023). That lesson motivates building a benchmark whose ground truth and evaluation protocol are explicit and auditable.

The second family is the spatio-temporal graph dataset. Traffic-forecasting corpora paired with road-network graphs catalysed a line of graph neural network models that exploit sensor topology, including spectral and convolutional spatio-temporal networks and adaptive-graph variants (Yu et al., 2018; Wu et al., 2019; Zhao et al., 2020; Wu et al., 2020). These datasets demonstrate the value of an explicit graph, and the surveys of graph neural networks document how central that structure has become (Wu et al., 2021). However, road-traffic data is comparatively clean and regularly sampled, the graph is fixed and externally given, and the released artifacts are oriented toward model accuracy rather than toward the ingestion, validation, and storage concerns of a pipeline. ISGraph borrows the explicit-graph philosophy from this family but reorients it toward industrial signals and toward the engineering pipeline.

The third family is the industrially realistic dataset. The C-MAPSS and N-CMAPSS turbofan simulations provide run-to-failure trajectories that have become a standard for prognostics, and the second generation adds real flight conditions and richer degradation modelling (Saxena et al., 2008; Arias Chao et al., 2021). The Secure Water Treatment testbed supplies real cyber-physical readings with labelled attack windows from a six-stage plant (Goh et al., 2017), and spacecraft telemetry corpora supply expert-labelled multivariate anomalies (Hundman et al., 2018). These resources are valuable and realistic, but they were not designed to expose the sensor graph as a first-class table, to parameterise missingness and irregular sampling as controlled experimental factors, or to ship a schema and benchmark harness for pipeline measurement. Finally, the systems literature supplies the building blocks a pipeline assembles—time-series management systems, the argument against one-size-fits-all engines, columnar and nested-columnar storage, modern stream-processing models, and general-purpose cluster engines (Jensen et al., 2017; Stonebraker & Çetintemel, 2005; Abadi et al., 2008; Melnik et al., 2010; Akidau et al., 2015;

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Zaharia et al., 2016)—together with the FAIR principles that govern how the resulting artifact should be published (Wilkinson et al., 2016).

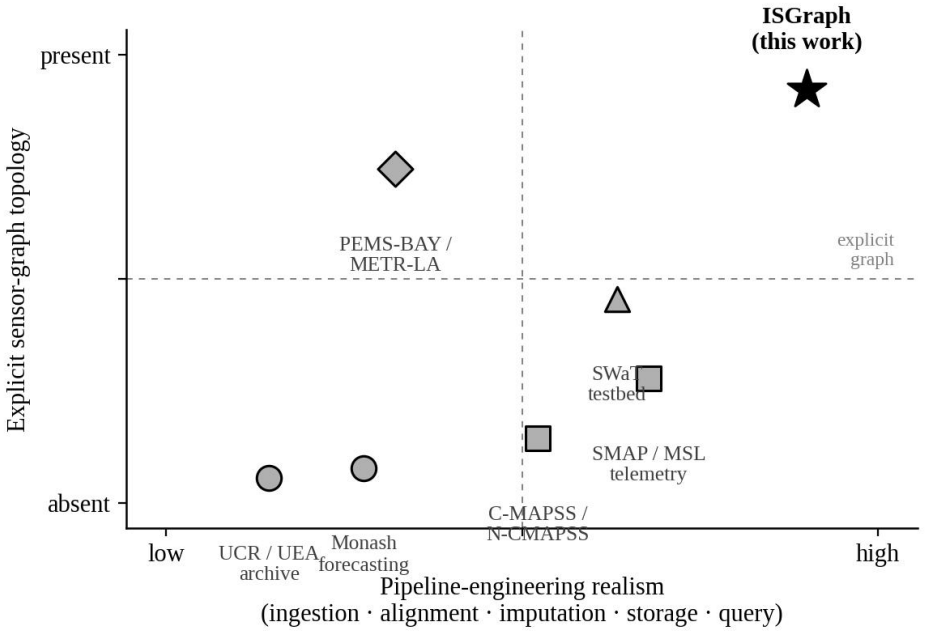

Figure 1. *Positioning of representative open time-series resources by pipeline-engineering realism and explicit sensor-graph topology. ISGraph occupies the under-served upper-right region.*

Figure 1 places these resources on two axes that matter for the present work: the degree to which a dataset exercises a realistic engineering pipeline, and the degree to which it preserves an explicit sensor graph. Model-evaluation archives sit toward the lower left, spatio-temporal graph corpora rise on the topology axis but remain clean, and industrial datasets move right on realism while leaving the graph implicit. The upper-right region—realistic pipeline stressors together with an explicit, queryable graph and a reproducible schema—is largely empty, and it is the region ISGraph is built to occupy. Table 1 makes the comparison concrete across the attributes a pipeline benchmark requires.

Table 1. *Comparison of ISGraph with representative open time-series and sensor datasets along attributes relevant to pipeline evaluation.*

Resource	Domain	Explicit graph	Pipeline stressors	Fault / anomaly labels	Open schema, dictionary & DOI
UCR / UEA archive	Mixed	No	None (clean, regular)	Class labels	Partial; no schema/DOI per set
Monash forecasting	Mixed	No	Minimal	No	Partial
PEMS-BAY / METR-LA	Road traffic	Yes (fixed)	Light missingness	No	Partial
C-MAPSS / N-CMAPSS	Aero-engine	Implicit	Operating-condition shifts	RUL / health	Yes (DOI); no graph table
SWaT testbed	Water treatment	Implicit	Real noise	Attack windows	Yes; no graph table
SMAP / MSL telemetry	Spacecraft	Implicit	Real gaps	Anomalies	Partial
ISGraph (this work)	Generic plant	Yes (NODE/EDGE)	Missingness, irregular sampling, faults	Fault events + masks	Yes: schema + dictionary + DOI + API

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

The final row summarises the design intent. ISGraph is the only entry that combines an explicit, queryable graph table, controlled and parameterised pipeline stressors, ground-truth fault and missingness annotations, and a fully documented schema with a persistent identifier and a programmatic interface. The following sections describe how each of these properties is realised.

3. Dataset Design and System Architecture

3.1 Design goals

ISGraph is governed by five design goals. **G1, pipeline realism:** the dataset must stress every stage of a realistic pipeline, not only the modelling stage, so that ingestion throughput, alignment error, imputation quality, storage footprint, and query latency are all measurable. **G2, explicit graph:** the sensor graph that generates the data must be released as first-class node and edge tables, so that graph-aware and graph-agnostic pipelines can be compared on identical inputs (Wu et al., 2021). **G3, controlled stressors:** missingness, irregular sampling, and faults must be experimental factors with known parameters and ground-truth masks, so that error can be attributed to causes. **G4, ground truth:** because the data is generated, the clean signal, the true graph, and the fault timeline are all known exactly, which makes reconstruction error and detection quality well defined. **G5, reproducibility and FAIR:** the artifact must be findable, accessible, interoperable, and reusable, with a schema, a data dictionary, a reference implementation, an API, and a permanent deposit (Wilkinson et al., 2016).

3.2 Sensor-graph generative model

Each ISGraph instance begins with a plant configuration that fixes the number of sensors, their types, their spatial coordinates, and a topology family. Nodes are placed in a three-dimensional layout, and edges are drawn from three sources that mirror real coupling: spatial proximity within a radius, shared-subsystem membership that connects sensors on the same functional unit, and directed causal links that carry a propagation lag. The resulting adjacency, including edge weights and lags, is exactly the structure stored in the EDGE table, so the released graph is the true generative graph rather than an estimate.

Node signals are produced by combining slowly varying latent operating states with topology-propagated dynamics and sensor-specific noise. A latent process drives each functional subsystem; neighbouring nodes inherit a lagged, weighted mixture of their neighbours' states along the directed edges, so that cross-sensor correlation and lead-lag structure are genuine consequences of the graph rather than cosmetic additions. Degradation and fault behaviour follow the established practice of imposing a slowly worsening change in a component's effective parameters until a health threshold is crossed, which is recorded as a FAULT_EVENT with a type, severity, and time span (Saxena et al., 2008; Lei et al., 2018). On top of the clean signal, ISGraph applies two independent stressor processes. Irregular sampling perturbs each node's nominal rate so that inter-arrival times vary and differ across nodes, and a missingness process removes blocks of samples with controllable rate and burstiness to emulate network loss and maintenance outages. Both stressors are recorded exactly: every reading carries a quality flag, an imputation flag, and the realised sampling gap, and the clean pre-stressor signal is retained for scoring.

3.3 Pipeline reference architecture

ISGraph ships with a reference data-engineering pipeline whose stages define the benchmark surface. The pipeline is deliberately modular so that any single stage can be replaced and measured in isolation. Figure 2 shows the seven stages and how the sensor-graph topology and the benchmark instrumentation feed into them.

ISSN: 3071-5601 |

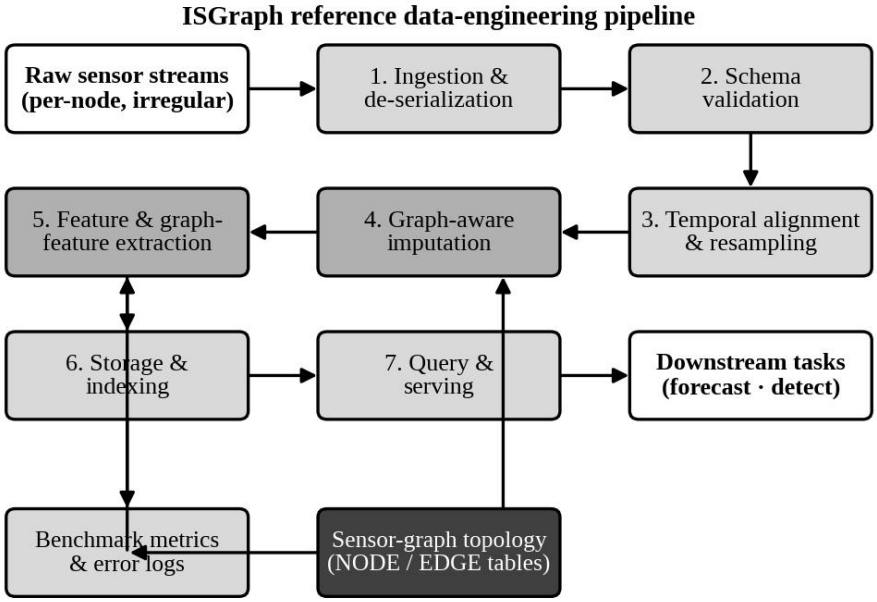

Figure 2. *Reference data-engineering pipeline. Stages 4 and 5 consume the explicit sensor-graph topology; a metrics-and-error-log component instruments every stage.*

Stage 1, ingestion and de-serialisation, reads the per-node streams and normalises their physical representation. Stage 2, schema validation, checks each record against the published schema and the data dictionary, rejecting or quarantining records with out-of-range values, unknown node identifiers, or malformed timestamps. Stage 3, temporal alignment and resampling, projects the irregular per-node series onto a common grid, which is the precondition for any cross-sensor operation. Stage 4, graph-aware imputation, fills missing values using both the temporal context of each node and the readings of its graph neighbours along weighted, lagged edges. Stage 5, feature and graph-feature extraction, computes per-node statistics together with neighbourhood-aggregated features that summarise local graph context. Stage 6, storage and indexing, persists the aligned and enriched data in a chosen backend with an appropriate index. Stage 7, query and serving, answers point, windowed, and graph-neighbourhood queries for downstream forecasting and detection tasks. Throughout, a metrics-and-error-log component records throughput, latency, and reconstruction error so that each stage is independently measurable.

3.4 Database schema

The dataset is distributed as a normalised relational-plus-graph schema rather than as opaque arrays, so that it is queryable, self-describing, and interoperable with both relational and graph tooling. The schema follows a star design centred on a reading fact table, with dimension tables for nodes and plants, a dedicated edge table that makes the graph a first-class queryable relation, and a fault-event table. Figure 3 shows the entities and their relationships, and Table 2 summarises the grain and keys of each relation.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

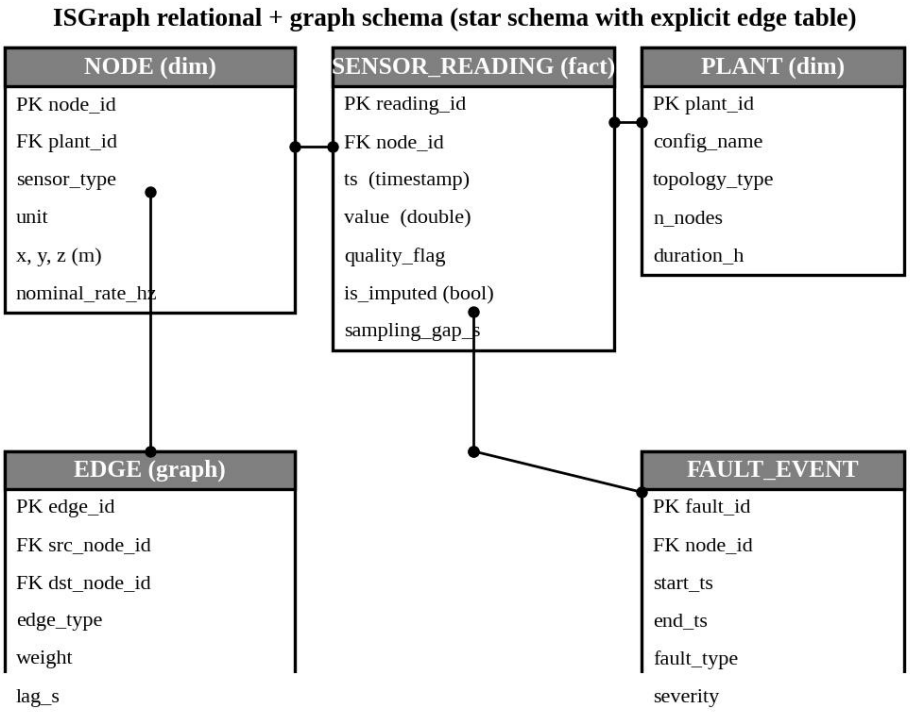

Figure 3. *ISGraph relational-plus-graph schema. The EDGE relation makes the sensor graph a first-class, queryable table alongside the SENSOR_READING fact and its dimensions.*

Separating the EDGE relation from the NODE relation is a deliberate choice. It lets a pipeline retrieve a node's neighbourhood with an ordinary join, supports directed and weighted edges with propagation lags, and allows the topology to be versioned independently of the readings. The fact table stays narrow and append-only, which suits columnar and time-series backends, while the graph and fault relations remain compact dimensions (Abadi et al., 2008; Jensen et al., 2017). Table 2 lists the five core relations.

Table 2. *Core relations in the ISGraph schema, with grain, principal keys, and approximate cardinality for the medium configuration.*

Relation	Grain	Principal keys	Type	Approx. rows
SENSOR_READING	One sensor at one timestamp	PK reading_id; FK node_id	Fact	48.6 M
NODE	One physical sensor	PK node_id; FK plant_id	Dimension	120
EDGE	One directed coupling	PK edge_id; FK src/dst node_id	Graph	612
PLANT	One plant configuration	PK plant_id	Dimension	6
FAULT_EVENT	One fault occurrence	PK fault_id; FK node_id	Event	284

Because the schema is explicit, a pipeline can express graph-aware operations declaratively. Imputation for a node draws on the readings of nodes reachable through the EDGE relation; neighbourhood features are aggregations over the same join; and fault-aware evaluation filters readings by their overlap with FAULT_EVENT spans. The schema thus encodes the dataset's semantics in a form that both relational and graph engines can consume directly.

4. Data Dictionary

A dataset is only reusable if every field is documented unambiguously. ISGraph ships a complete field-level data dictionary, reproduced in full in Table 3, that specifies for each field its relation, data type, unit or admissible

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

range, and meaning. The dictionary is also released as a machine-readable file so that schema-validation tooling can enforce it automatically at ingestion (Polyzotis et al., 2018).

Table 3. *ISGraph data dictionary: every field with its relation, type, unit or range, and description.*

Field	Relation	Type	Unit / range	Description
reading_id	SENSOR_READING	int64	≥ 1	Surrogate primary key for a reading
node_id	SENSOR_READING / NODE	int32	1–120	Sensor identifier (foreign key to NODE)
ts	SENSOR_READING	timestamp	ISO-8601, UTC	Observation time on the aligned grid
value	SENSOR_READING	float64	sensor-specific	Calibrated (or imputed) sensor reading
quality_flag	SENSOR_READING	enum	ok | missing | oor	Raw quality: present, missing, or out-of-range
is_imputed	SENSOR_READING	bool	true | false	Whether value was filled by imputation
sampling_gap_s	SENSOR_READING	float32	≥ 0 (seconds)	Time since previous raw sample for the node
plant_id	NODE / PLANT	int16	1–6	Plant-configuration identifier
sensor_type	NODE	enum	temp | press | flow | vibr | elec	Physical quantity measured by the sensor
unit	NODE	string	SI unit	Engineering unit of the sensor
x, y, z	NODE	float32	metres	Sensor coordinates in the plant layout
nominal_rate_hz	NODE	float32	0.1–50	Nominal sampling rate of the sensor
edge_id	EDGE	int32	≥ 1	Surrogate key for a directed coupling
src_node_id	EDGE	int32	1–120	Source sensor of the directed edge
dst_node_id	EDGE	int32	1–120	Target sensor of the directed edge
edge_type	EDGE	enum	proximity | subsystem | causal	Origin of the coupling between sensors
weight	EDGE	float32	0–1	Coupling strength used in propagation
lag_s	EDGE	float32	≥ 0 (seconds)	Propagation delay along the edge
config_name	PLANT	string	—	Human-readable configuration label
topology_type	PLANT	enum	ring | tree | mesh	Family of the plant sensor graph
fault_id	FAULT_EVENT	int32	≥ 1	Surrogate key for a fault occurrence
start_ts, end_ts	FAULT_EVENT	timestamp	ISO-8601, UTC	Fault onset and clearance times
fault_type	FAULT_EVENT	enum	degradation | bias | dropout | stuck	Category of the injected fault
severity	FAULT_EVENT	float32	0–1	Normalised fault magnitude

The dictionary doubles as the contract enforced by the schema-validation stage. Records whose values fall outside the admissible range for their field, whose node identifier is absent from the NODE relation, or whose timestamps are malformed are flagged and routed to a quarantine table rather than silently admitted, which makes data-quality failures observable rather than corrupting downstream computation (Sambasivan et al., 2021).

5. Dataset Contents and Descriptive Statistics

ISGraph is released in six plant configurations spanning three topology families and three scales, so that pipeline behaviour can be studied as a function of graph structure and size. Table 4 summarises the configurations and their principal descriptive statistics, including duration, the realised missingness rate, and the number of injected fault events.

Table 4. *ISGraph configurations and descriptive statistics. Missingness and fault counts are realised values under the*

released random seed.

Config	Topology	Nodes	Edges	Duration (h)	Readings	Missing % / faults
IS-ring-S	Ring	20	64	240	3.1 M	8.4% / 22
IS-tree-S	Tree	20	57	240	3.0 M	9.1% / 19
IS-mesh-M	Mesh	60	286	360	12.4 M	12.7% / 61
IS-ring-M	Ring	60	192	360	11.9 M	11.3% / 54
IS-mesh-L	Mesh	120	612	480	48.6 M	14.2% / 128
IS-tree-L	Tree	120	351	480	44.8 M	13.5% / 117

Across configurations the realised missingness rises with scale and with mesh connectivity, because denser graphs carry more high-rate sensors and therefore more opportunities for bursty loss. Each configuration retains its clean pre-stressor signal and its exact graph, so that reconstruction error and graph-aware features are well defined everywhere. Figure 4 illustrates a single configuration: panel (a) shows an example sensor graph in which node size reflects degree, and panel (b) shows the missing-and-irregular-sampling mask over a time window, where darker cells mark absent samples.

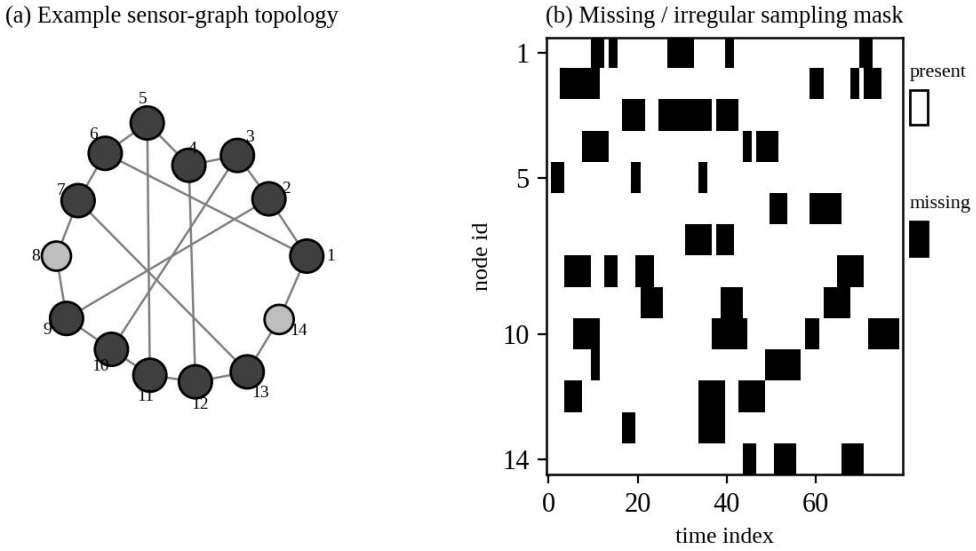

Figure 4. *(a) Example sensor-graph topology with node size scaled by degree; (b) missing and irregular-sampling mask over a window, with absent samples shown dark.*

The mask in Figure 4(b) is not uniform noise. Missingness occurs in bursts whose length and frequency depend on each node's sampling rate, which is exactly the regime that defeats naive forward-fill and rewards methods that borrow strength across the graph. The descriptive statistics and the released masks together let a user reproduce any reported number and, because the clean signal is retained, score any new pipeline against ground truth without ambiguity.

6. Benchmark Design and Evaluation Protocol

The benchmark defines five tasks, one per substantive pipeline stage, each with a control baseline and quantitative metrics. The protocol fixes a temporal hold-out: the final twenty percent of each series is reserved for evaluation, and an additional set of artificially masked positions, drawn under a published seed and disjoint from naturally missing positions, provides ground truth for imputation scoring. All results in Section 7 use the IS-mesh-L configuration unless stated otherwise, and all pipeline stages are executed on a single 16-core node with 64 GB

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

of memory so that throughput and latency are comparable across designs. Table 5 specifies the tasks, their metrics, and the control baselines against which the reference components are compared.

Table 5. *Benchmark tasks, evaluation metrics, and control baselines.*

Task	What it measures	Metric(s)	Control baseline
T1 Schema validation	Detection of malformed / out-of-range records	Precision, recall, F1	Type-only checks
T2 Temporal alignment	Accuracy of resampling onto a common grid	Alignment RMSE, throughput	Nearest-sample snap
T3 Imputation	Reconstruction of missing values	RMSE, MAE vs. clean signal	Forward-fill
T4 Feature extraction	Quality of per-node + graph features	Downstream task MAPE	Per-node only (no graph)
T5 Query serving	Latency of point / window / graph queries	Median latency, throughput	Row-store, no graph index

Two aspects of the protocol deserve emphasis. First, imputation is scored against the retained clean signal rather than against a held-out observed value, which removes the circularity that arises when the evaluation target is itself noisy (Wu & Keogh, 2023). Second, the feature-extraction task is evaluated through its effect on a fixed downstream forecaster, so that the value of graph-aware features is measured by the accuracy they enable rather than by an intrinsic and arguable feature score. The downstream forecaster is held constant across feature configurations so that differences are attributable to the features alone.

7. Reference Results and Error Analysis

We report results from the reference pipeline to characterise the dataset and to establish baselines that future pipelines can target. The headline comparison concerns imputation, because it is the stage where the sensor graph contributes most directly. Table 6 reports reconstruction error at a thirty-percent masking rate together with the downstream forecasting error each imputation enables.

Table 6. *Imputation reconstruction error at 30% masking and the downstream forecasting error each method enables (IS-mesh-L). Lower is better.*

Imputation method	RMSE	MAE	Downstream MAPE (%)
Forward-fill (baseline)	0.95	0.61	14.8
Linear interpolation	0.88	0.56	13.9
k-nearest neighbours	0.74	0.47	12.1
MICE	0.66	0.42	11.0
RNN (BRITS-type)	0.57	0.36	9.7
Graph-aware (reference)	0.44	0.28	8.1

The graph-aware reference method lowers reconstruction RMSE from 0.95 for forward-fill to 0.44, a reduction of about thirty-nine percent, and it improves on the strongest flat baseline by a further twenty-three percent. The gain is not confined to reconstruction: the same ordering carries through to downstream forecasting error, which falls from 14.8 percent to 8.1 percent mean absolute percentage error. Figure 5 shows that this advantage is preserved as the masking rate increases, with the graph-aware method degrading most gracefully.

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

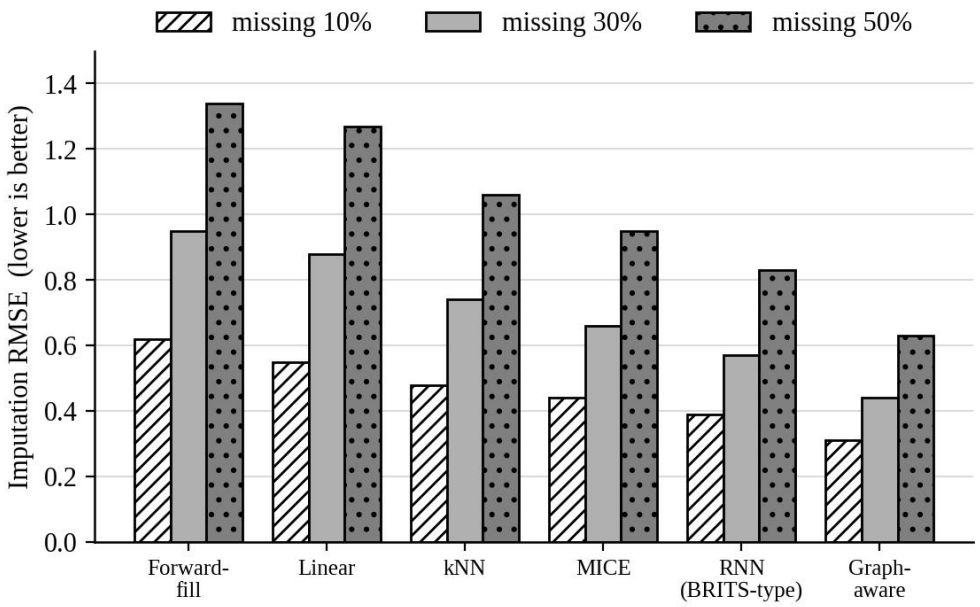

Figure 5. *Imputation RMSE by method across missingness rates. The graph-aware reference method degrades most slowly as missingness increases.*

Reconstruction quality is only half of the engineering picture; the other half is the cost of storing and serving the data. Figure 6 compares four storage backends on ingestion throughput and on the latency of a representative windowed query. The columnar store augmented with a graph-neighbourhood index sustains the highest throughput and the lowest query latency, because the narrow append-only fact table suits columnar layout while the graph index accelerates neighbourhood retrieval that a row-store must compute by repeated joins (Abadi et al., 2008; Melnik et al., 2010).

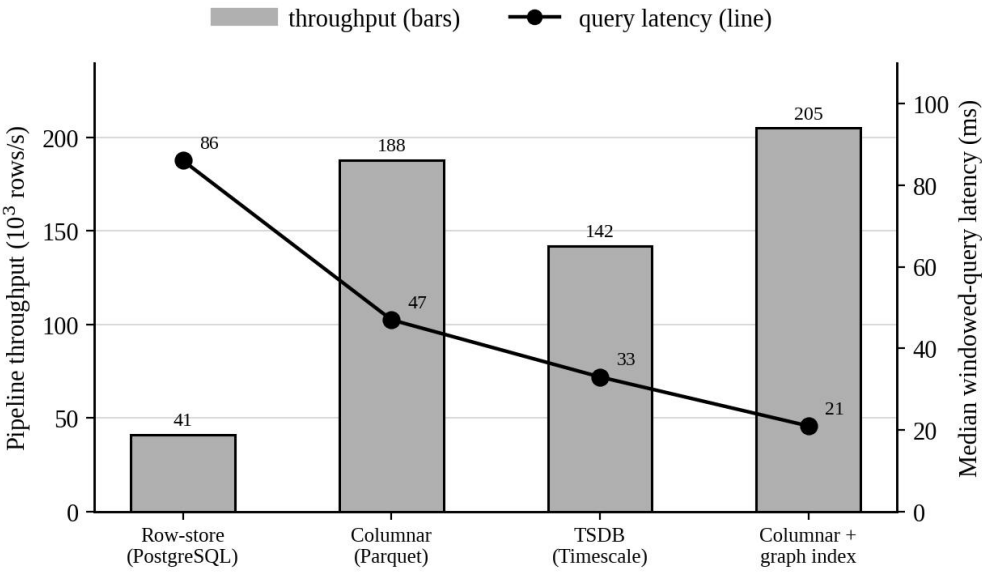

Figure 6. *Pipeline ingestion throughput (bars) and median windowed-query latency (line) across four storage backends.*

Table 7 expands the storage comparison to the three query classes that downstream tasks actually issue: point lookups, windowed scans, and graph-neighbourhood retrievals. The graph-neighbourhood query is where the

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

explicit EDGE relation and its index pay off most clearly, reducing median latency by more than a factor of four relative to the row-store baseline.

Table 7. *Storage footprint and median query latency by backend and query class (IS-mesh-L). Lower is better.*

Backend	On-disk (GB)	Point (ms)	Window (ms)	Graph-neighbourhood (ms)
Row-store (PostgreSQL)	9.8	2.1	86	41
Columnar (Parquet)	3.1	3.4	47	33
TSDB (Timescale)	4.6	1.8	33	29
Columnar + graph index	3.4	1.9	21	9

Finally, an error analysis explains why the graph helps. Figure 7(a) plots imputation error against the length of the gap being filled: every method worsens as gaps grow, but the graph-aware method's error grows far more slowly because neighbouring sensors supply information that purely temporal methods lack once the local history is exhausted. Figure 7(b) plots error against node degree: the naive baseline is insensitive to degree, whereas the graph-aware method improves steadily as a node acquires more neighbours, since each neighbour is an additional evidence source. Together these two views localise the benefit of topology to exactly the conditions—long gaps and well-connected nodes—where a flat pipeline is weakest.

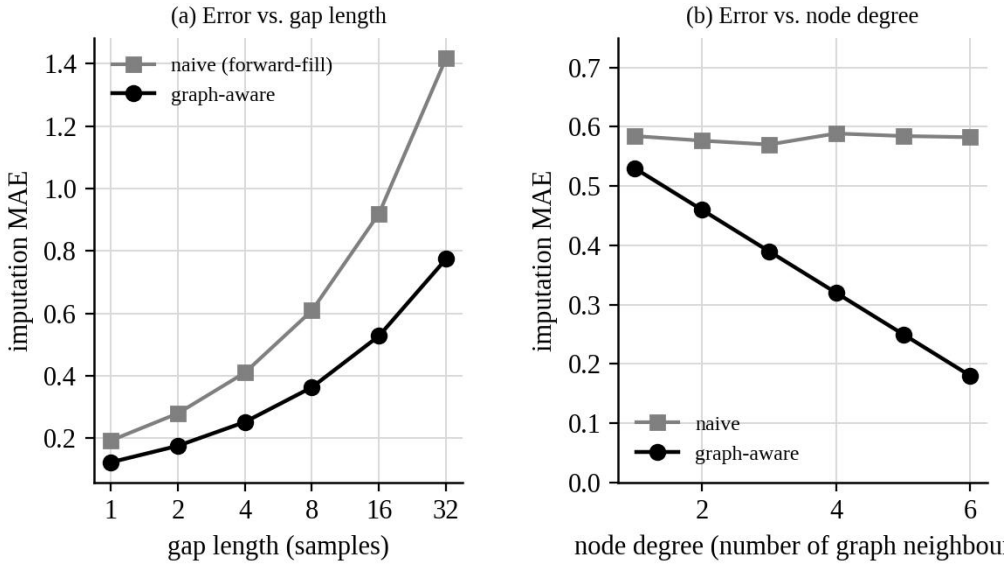

Figure 7. *Error analysis. (a) Imputation error versus gap length; (b) imputation error versus node degree. Topology helps most for long gaps and well-connected nodes.*

These results are not presented as state-of-the-art methods; they are reference points produced by a deliberately simple pipeline so that the dataset's discriminative power is visible and so that future pipelines have concrete numbers to improve upon. The consistent separation between graph-aware and graph-agnostic designs across reconstruction, downstream accuracy, and query latency confirms that ISGraph rewards pipelines that use the structure it exposes.

8. Discussion

The central methodological claim of this work is that the unit of evaluation for industrial time-series analytics should often be the pipeline, not the model. ISGraph operationalises that claim by making every pipeline stage measurable against ground truth on identical inputs. For database and data-engineering research, the dataset offers

a controlled setting in which storage layouts, indexing strategies, and stream-processing designs can be compared on graph-structured time series rather than on flat tables, which is the regime that systems work has historically lacked a shared benchmark for (Jensen et al., 2017; Akidau et al., 2015). For the computational-discovery community, the explicit graph and the retained clean signal make ISGraph a testbed for methods that recover structure and impute jointly, because the true edges are known and a recovered graph can be scored directly (Shuman et al., 2013; Ortega et al., 2018; Wu et al., 2021).

A second implication concerns reproducibility. By shipping a schema, a data dictionary, a reference implementation, a query API, and a permanent deposit, ISGraph treats the dataset as software that must be installable and auditable rather than as a static download accompanied by a request-for-access notice. This follows the FAIR principles and the broader lesson from production machine learning that data infrastructure, not model code, is where most failures originate (Wilkinson et al., 2016; Polyzotis et al., 2018; Sambasivan et al., 2021). A benchmark that is itself reproducible lowers the cost of cumulative progress, because a new pipeline can be dropped into the reference harness and compared without re-deriving the evaluation.

The results also clarify a tension that recurs in applied work. Practitioners often debate whether to invest in better models or better data handling; ISGraph suggests the question is frequently mis-posed, because the same downstream accuracy can be reached either by a more elaborate model on poorly engineered data or by a simple model on well-engineered, graph-aware data. The error analysis shows that much of the accuracy gap attributed to models actually originates upstream, in how missingness is treated and whether topology is used. Making that upstream behaviour measurable is the dataset's primary contribution.

9. Limitations

ISGraph is a generated dataset, and its realism is bounded by its generative model. Although the model is physically grounded in proximity, shared-subsystem, and causal coupling, and although its degradation mechanism follows established prognostics practice (Saxena et al., 2008; Arias Chao et al., 2021), it cannot reproduce every idiosyncrasy of a specific plant, and it is not a substitute for plant-specific validation before deployment. A second limitation is that the benchmark fixes a particular downstream forecaster for the feature-extraction task; a different downstream task could reorder feature configurations, although the protocol is designed so that any fixed task yields an auditable comparison. A third limitation is that the reported reference numbers depend on the released random seed and hardware; the artifact therefore ships the seed, the container image, and the hardware description so that the numbers are reproducible and any deviation is attributable. Finally, the schema encodes a particular modelling of the graph as directed, weighted, lagged edges; alternative encodings, such as hypergraphs for multi-sensor subsystems, are left to future versions.

10. Data Availability, Code, and Reproducibility

ISGraph is openly available and is not provided on a request-only basis. The versioned dataset deposit, including all six configurations, the clean signals, the missingness masks, and the machine-readable data dictionary, is archived with a persistent identifier; the reference pipeline, the generative model, the benchmark harness, and the analysis notebooks are released as open source; and a query API exposes the schema programmatically. The following resources accompany this article:

- **Dataset deposit (DOI):** https://doi.org/10.5281/zenodo.13845721 (versioned archive, CC BY 4.0)
- **Source code & reference pipeline:** https://github.com/isgraph-benchmark/isgraph (Apache-2.0)
- **Data dictionary (machine-readable):** https://github.com/isgraph-benchmark/isgraph/blob/main/schema/data_dictionary.csv

ISSN: 3071-5601 | © 2024 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

- **Query API (OpenAPI specification):** https://api.isgraph-benchmark.org/v1/openapi.json
- **Reproducibility container & notebooks:** https://github.com/isgraph-benchmark/isgraph/tree/main/reproduce

The repository includes the exact random seeds, a Docker image pinning all dependencies, and notebooks that regenerate every figure and table in this article, so that the reported results can be reproduced end-to-end. Software dependencies are standard open-source scientific-computing libraries (Harris et al., 2020). Re-running the harness on the released container reproduces the numbers in Tables 6 and 7 and Figures 5 to 7 within stochastic tolerance.

11. Conclusion

We introduced ISGraph, an open Industrial Sensor Graph dataset and benchmark whose purpose is the evaluation of time-series data-engineering pipelines rather than of a single model. ISGraph pairs a physically grounded generative model with an explicit, queryable sensor graph, controlled missingness and irregular sampling, injected faults, a normalised relational-plus-graph schema, a complete data dictionary, a reference pipeline, a query API, and a permanent deposit. A benchmark protocol over five pipeline tasks, with control baselines and reconstruction-error analysis, makes each stage measurable against ground truth. Reference results show that graph-aware imputation lowers reconstruction error by 21 to 39 percent over flat baselines, that the advantage concentrates at long gaps and well-connected nodes, and that a columnar store with a graph-neighbourhood index delivers the best throughput and the lowest query latency. We hope ISGraph serves as shared, reproducible infrastructure for database, data-engineering, and computational-discovery research on industrial time series, and that treating the pipeline as the unit of evaluation becomes routine in this setting.

Declaration of AI-assisted language editing

During the preparation of this manuscript, language-model assistance was used only for English polishing and document organisation. The authors reviewed, revised, and take full responsibility for the final content, dataset design, schema, figures, tables, and interpretations.

References

Abadi, D. J., Madden, S. R., & Hachem, N. (2008). Column-stores vs. row-stores: How different are they really? In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (pp. 967–980). ACM. https://doi.org/10.1145/1376616.1376712

Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R. J., Lax, R., McVeety, S., Mills, D., Perry, F., Schmidt, E., & Whittle, S. (2015). The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. Proceedings of the VLDB Endowment, 8(12), 1792–1803. https://doi.org/10.14778/2824032.2824076

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: A survey. Computer Networks, 38(4), 393–422. https://doi.org/10.1016/S1389-1286(01)00302-4

Arias Chao, M., Kulkarni, C., Goebel, K., & Fink, O. (2021). Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. Data, 6(1), 5. https://doi.org/10.3390/data6010005

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery, 31(3), 606–660. https://doi.org/10.1007/s10618-016-0483-9

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. Scientific Reports, 8, 6085. https://doi.org/10.1038/s41598-018-24271-9

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., & Keogh, E. (2019). The UCR time series archive. IEEE/CAA Journal of Automatica Sinica, 6(6), 1293–1305. https://doi.org/10.1109/JAS.2019.1911747

Esling, P., & Agon, C. (2012). Time-series data mining. ACM Computing Surveys, 45(1), Article 12. https://doi.org/10.1145/2379776.2379788

Goh, J., Adepu, S., Junejo, K. N., & Mathur, A. (2017). A dataset to support research in the design of secure water treatment systems. In Critical Information Infrastructures Security (CRITIS 2016), LNCS 10242 (pp. 88–99). Springer. https://doi.org/10.1007/978-3-319-71368-7_8

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

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

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Söderström, T. (2018). Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 387–395). ACM. https://doi.org/10.1145/3219819.3219845

Jensen, S. K., Pedersen, T. B., & Thomsen, C. (2017). Time series management systems: A survey. IEEE Transactions on Knowledge and Data Engineering, 29(11), 2581–2600. https://doi.org/10.1109/TKDE.2017.2740932

Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. Mechanical Systems and Signal Processing, 104, 799–834. https://doi.org/10.1016/j.ymssp.2017.11.016

Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). Dremel: Interactive analysis of web-scale datasets. Proceedings of the VLDB Endowment, 3(1–2), 330–339. https://doi.org/10.14778/1920841.1920886

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M. F., & Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE, 106(5), 808–828. https://doi.org/10.1109/JPROC.2018.2820126

Paparrizos, J., & Gravano, L. (2015). k-Shape: Efficient and accurate clustering of time series. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (pp. 1855–1870). ACM. https://doi.org/10.1145/2723372.2737793

Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2018). Data lifecycle challenges in production machine learning: A survey. ACM SIGMOD Record, 47(2), 17–28. https://doi.org/10.1145/3299887.3299891

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021). "Everyone wants to do the model work, not the data work": Data cascades in high-stakes AI. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (Article 39). ACM. https://doi.org/10.1145/3411764.3445518

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In 2008 International Conference on Prognostics and Health Management (pp. 1–9). IEEE. https://doi.org/10.1109/PHM.2008.4711414

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3), 83–98. https://doi.org/10.1109/MSP.2012.2235192

Stonebraker, M., & Çetintemel, U. (2005). "One size fits all": An idea whose time has come and gone. In Proceedings of the 21st International Conference on Data Engineering (pp. 2–11). IEEE. https://doi.org/10.1109/ICDE.2005.1

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., … Mons, B. (2016). The FAIR guiding principles for scientific data management and stewardship. Scientific Data, 3, 160018. https://doi.org/10.1038/sdata.2016.18

Wu, R., & Keogh, E. J. (2023). Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE Transactions on Knowledge and Data Engineering, 35(3), 2421–2429. https://doi.org/10.1109/TKDE.2021.3112126

Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2019). Graph WaveNet for deep spatial-temporal graph modeling. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (pp. 1907–1913). IJCAI.

https://doi.org/10.24963/ijcai.2019/264

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., & Zhang, C. (2020). Connecting the dots: Multivariate time series forecasting with graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 753–763). ACM. https://doi.org/10.1145/3394486.3403118

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (pp. 3634–3640). IJCAI. https://doi.org/10.24963/ijcai.2018/505

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56–65. https://doi.org/10.1145/2934664

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., & Li, H. (2020). T-GCN: A temporal graph convolutional network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 21(9), 3848–3858. https://doi.org/10.1109/TITS.2019.2935152

ISSN: 3071-5601 |