Streaming Commodity Forecasting with Kolmogorov–Arnold Networks on the Edge

1st Dat Le
School of Information Technology
Deakin University
Geelong, VIC 3220, Australia
d.le@deakin.edu.au

2nd Sutharshan Rajasegarar
School of Information Technology
Deakin University
Geelong, VIC 3220, Australia
sutharshan.rajasegarar@deakin.edu.au

3rd Wei Luo
School of Information Technology
Deakin University
Geelong, VIC 3220, Australia
wei.luo@deakin.edu.au

4th Thanh Thi Nguyen
Faculty of Information Technology
Monash University
Clayton, VIC 3800, Australia
thanh.nguyen9@monash.edu

5th Minh Ngoc Dinh
School of Science,Engineering,Tech
RMIT University Vietnam
Ho Chi Minh City, 70000, Vietnam
minh.dinh4@rmit.edu.vn

6th Maia Angelova
Aston Digital Futures Institute
Aston University
Birmingham B4 7ET, United Kingdom
m.angelova@aston.ac.uk

***Abstract*—This paper introduces a lightweight, reproducible pipeline for real-time commodity price forecasting at the network edge, combining a Kafka-based streaming loop with compact neural predictors. Using Kolmogorov–Arnold Networks (KAN) alongside LSTM and TS-Mixer baselines, the system performs one-step-ahead inference on futures contract for West Texas Intermediate (WTI) crude oil, gold, silver, and copper. KAN consistently achieves the highest accuracy across all assets while maintaining low computational overhead. Importantly, end-to-end latency is dominated by networking and coordination, not by the model computation overhead, highlighting a key system-level insight for edge deployment. The pipeline runs entirely on a single local machine without cloud services, and is fully reproducible, offering a practical and scalable solution for near-real-time commodity forecasting.**

***Index Terms*—Commodity forecasting, Streaming analytics, Kolmogorov–Arnold Networks, Edge machine learning, Docker, Kafka.**

I. Introduction

Commodities markets today operate as highly interconnected, large-scale networks where millions of transactions and price updates propagate in real time across global exchanges [1]. Traditional batch forecasting pipelines struggles to keep pace with this velocity, creating bottlenecks between data ingestion and actionable insights [2].

Accurate and timely forecasting of commodity prices underpins decision-making in energy, manufacturing, and financial markets [3]. As data pipelines migrate from batch processing to online streams, the latency that matters to users increasingly reflects system effects—network I/O, queueing, and consumer coordination—rather than only model computation [4]. In this setting, small models that run entirely on local hardware are attractive, as they are easier to deploy as well as control, and less vulnerable than cloud-heavy stacks, yet they still require to deliver competitive accuracy under streaming constraints.

Despite rapid progress in sequence modeling, several practical gaps remain [5]. First, most studies evaluate models offline on static train/test splits and report inference time in isolation, leaving a disconnect between reported latency and the end-to-end delay observed in real systems. Second, implementations that are lightweight and "edge-ready" are scarce; many require customised infrastructure or cloud services, limiting reproducibility on laptops or lab machines. Finally, multi-asset streaming introduces contention and interleaving effects rarely studied in isolation, yet common in practice when several instruments/stocks are considered concurrently.

In order to address these limitations, this paper proposes a compact, local, Kafka-based forecasting loop that ingests daily ticks (minimum price movements for a commodity), maintains a sliding window, and emits one-step-ahead predictions in real time. The system compares four compact forecasters under identical conditions including olmogorov–Arnold Network (KAN), Long Short-Term Memory network (LSTM), and Time-Series Mixer (TS-Mixer), using the same pre-processing, window size, horizon, optimizer family, device configuration, and evaluation protocol. Instrumentation records two complementary notions of time: (i) model-only inference per tick and (ii) the true end-to-end latency from publish to consume to forecast. This design isolates model behavior while exposing the operational costs that dominate user-visible delay.

The evaluation targets multiple commodities to reflect realistic portfolio monitoring: energy (West Texas Intermediate (WTI) crude oil), precious and industrial metals (gold, silver, copper), with the pipeline readily extensible to natural gas and other assets. Results show that compact KAN models achieve strong average accuracy across assets while keeping computation lightweight on CPU. At the same time, measurements indicate that end-to-end latency is governed primarily by networking and coordination in the streaming stack, making model computation a minor fraction of the total delay perceived by end users.

This paper has three main contributions as follows:

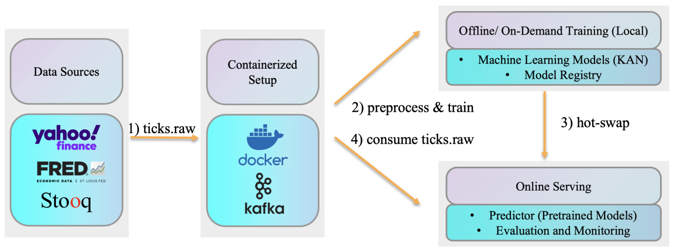

Fig. 1. The pipeline runs end-to-end on a local Docker host. First, the ingestor fetches market quotes from the data sources (Yahoo Finance, Stooq, optional FRED) and produces JSON ticks—asset, price, ts—to Kafka topic: ticks.raw (1). For model building, the Offline/On-Demand Training path batch-exports from ticks.raw, then preprocesses (clean, align, split), scales + windows the series, and trains compact models (KAN/LSTM/TSMixer) (2). The fitted artifacts—model.pt, scaler.pkl, manifest.json—are written to the Model Registry, and the Online Serving process loads them on start or hot-swaps when retrained (3). During live operation, the predictor consumes ticks.raw, keeps a per-asset ring buffer (WINDOW), applies the saved scaler, runs inference (KAN by default), and publishes predictions with timestamps to topic: preds.out (4). A lightweight evaluator joins the next truth from ticks.raw with preds.out to compute MAPE/RMSE/MAE and latency (inference ms, end-to-end ms), enabling continuous, transparent monitoring of accuracy and responsiveness on a single machine.

- **Edge-ready streaming pipeline & deployment:** A fully reproducible, edge-ready forecasting pipeline that integrates compact neural models with a realistic Kafka-based streaming loop, enabling near-real-time commodity price prediction without cloud infrastructure.
- **Compact KAN with fair, online benchmarks:** A systematic evaluation of Kolmogorov–Arnold Networks (KAN) as compact, CPU-efficient forecasters, demonstrating superior accuracy across four major commodities compared to LSTM and TS-Mixer under identical conditions.
- **Networking-aware evaluation & reproducibility:** A novel latency-aware benchmarking framework that decomposes end-to-end delay into model compute and system overhead, revealing that networking and coordination—not model complexity—are the dominant factors in real-time responsiveness.

II. Related Work

Prior studies consider pipeline design, model accuracy, and deployment constraints, but often treat these aspects separately. This section reviews the factors and the common assumptions that influence real-time forecasting for commodity futures (energy and metals).

A. Streaming time–series analytics.

Production forecasting has been shifting from batch jobs to online streaming on message buses (e.g., Kafka [6], [7]) with stream processors (e.g., Flink/Spark Streaming) [8]. In commodities, this shift reflects the need to combine daily settlements and intraday ticks around exchange calendars, contract rolls, and macro news releases. Systems related literature consider optimizing throughput, back-pressure, and fault-tolerance. Forecasting related works aim to evaluate offline, end-to-end per-tick latency (ingest $\rightarrow$ predict $\rightarrow$ next-tick validation) and their decomposition into network, coordination, and computation in a reproducible setup [9].

B. Neural forecasters for financial data.

Long Short-Term Memory (LSTM) models remain common for one-step commodity forecasting due to their sequential inductive bias and robustness under limited context [10], [11]. TS-Mixer, inspired by MLP-Mixer, replaces recurrence/attention with lightweight time–channel mixing, yielding competitive accuracy and very low CPU inference cost, which are useful for edge deployments considering daily futures for oil, gas, and metals [12]. Within a Kafka loop [6], these classes can be trained offline and hot-swapped online for per-tick accuracy/latency evaluation across assets.

C. Kolmogorov–Arnold Networks (KAN).

Kolmogorov–Arnold Networks (KAN) have been proposed as compact universal approximators using additive compositions and kernel/spline-like bases [13], [14]. Their small parameter budgets are appealing for edge inference on volatile commodities where noise, seasonality (e.g., winter gas demand), and structural breaks (e.g., supply shocks) penalize heavy models. Existing KAN related work is largely offline and on small regression suites [15], [16], while systematic, online comparisons against LSTM and TS-Mixer in a *streaming* commodity setting with identical preprocessing and per-tick scoring remain limited [17].

D. Edge inference and networking constraints.

Edge-centric machine learning (ML) emphasizes shrinking models (distillation, pruning, quantization), batching, and operator fusion to meet CPU/NPU latency and energy budgets [18]. For real-time commodity data, networking-aware ML needs to account for transport overhead, serialization formats, consumer-group coordination, and topic/partition design [19]. Few studies consider a lightweight forecaster with a realistic producer/consumer loop (including next-tick truth joining across futures calendars) and report both inference time and end-to-end latency under a single, reproducible protocol.

E. Data sourcing and reproducibility.

Commodity studies often rely on a single public source (e.g., Yahoo Finance) or proprietary feeds, risking rate-limits, silent gaps, and roll-over inconsistencies across front-month contracts [20]. Reproducibility improves with documented fallbacks (e.g., Stooq for end of day (EOD) stock, Federal Reserve Economic Data (FRED) for macro series) [21], [22], deterministic windowing, explicit holiday calendars, and a model registry that pins scalers and artifacts per asset for repeatable edge runs.

F. Commodities forecasting.

Forecasting in commodity markets has long been studied due to their economic importance and inherent volatility. Energy commodities, such as crude oil and natural gas are strongly influenced by geopolitical risks, Organization of the

Petroleum Exporting Countries (OPEC) decisions, weather shocks, and inventory reports, making short-term price forecasting a key challenge [23]. Metals such as gold, silver, and copper, meanwhile, are driven by macroeconomic cycles, currency fluctuations, and safe-haven demand [12], [24]. More recently, machine learning approaches including LSTM and Mixer-style networks have been introduced to capture complex temporal patterns in commodity data, showing improved accuracy compared to classical baselines [25]. Despite these advances, most prior work remains focused on offline, batch evaluations. Studies that integrate commodity forecasting with streaming infrastructures and edge-ready models remain scarce, limiting applicability in real-time decision-making contexts, such as trading, risk management and hedging.

In summary, the literature on streaming analytics, neural forecasters, and edge deployment provides ingredients for real-time commodity prediction, but rarely integrates them end-to-end. This work addresses that gap by embedding a compact KAN forecaster in a Kafka-based pipeline and benchmarking per-tick accuracy and latency across key energy and metals assets under identical conditions.

III. Methodology

This work develops a compact and reproducible forecasting pipeline that integrates Kolmogorov–Arnold Networks with a real-time streaming infrastructure. The system is designed to operate on a local machine, containerized using Docker, and orchestrated through Apache Kafka to handle continuous price ticks. The primary contribution lies in demonstrating the efficiency of KAN in an edge-friendly streaming environment.

A. Problem Formulation

Let $\{p_t\}_{t=1}^{T}$ denotes a univariate commodity price series (e.g., WTI crude oil, gold, silver, copper). The goal is to predict the next tick p_{t+1} based on the past W observations:

$$\mathbf{x}_t = [p_{t-W+1}, \dots, p_t], \quad \hat{p}_{t+1} = f_\theta(\mathbf{x}_t), \tag{1}$$

where f_θ is the forecasting model parameterized by θ. In this study, f_θ is instantiated by KAN, and baseline forecasters (LSTM, TS-Mixer) are used for comparative evaluation.

B. Data Preprocessing

Historical data are retrieved from multiple public sources (Yahoo Finance, Stooq, and optionally FRED for robustness). Each series is cleaned, aligned, and resampled to daily frequency. A Min–Max scaler is then applied:

$$p_t^{\text{scaled}} = \frac{p_t - p_{\min}}{p_{\max} - p_{\min}}, \tag{2}$$

ensuring values remain within $[0, 1]$. Supervised learning pairs $(\mathbf{x}_t, p_{t+1})$ are constructed via a sliding window of size W.

C. Kolmogorov–Arnold Networks (KAN)

KAN [26] builds on the Kolmogorov–Arnold representation theorem, which states that any multivariate continuous function can be expressed as a sum of univariate functions. This property allows KAN to approximate nonlinear mappings with compact parameterization. Given input $\mathbf{x}_t$, the model predicts:

$$\hat{p}_{t+1} = \sum_{i=1}^{H} \phi_i(\mathbf{x}_t; \theta_i), \tag{3}$$

where ϕ_i are kernelized spline functions with adaptive knots, and H is the hidden dimension. Unlike recurrent or attention-based models, KAN relies on local function approximations that reduce complexity and improve efficiency on CPU-only setups. This makes it particularly well suited for edge devices and local streaming deployments.

D. Streaming Integration with Kafka and Docker

To evaluate models in a realistic streaming environment, the pipeline is fully containerized with Docker [27] and connected through Kafka [28] topics:

- **Producer:** Continuously emits normalized price ticks in JSON format (`asset`, `value`, `timestamp`) to the Kafka topic `ticks.raw`.
- **Consumer:** Maintains a ring buffer of the last W ticks per asset, retrieves the trained KAN model and corresponding scaler from a local model registry, and performs online inference.

The Dockerized setup ensures reproducibility across machines, while Kafka decouples data ingestion, inference, and evaluation, allowing realistic measurement of delays caused by networking and coordination in addition to model computation.

E. Training and Loss Function

Offline training is performed using the Mean Squared Error (MSE) loss:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (p_i - \hat{p}_i)^2, \tag{4}$$

where N is the number of training samples. Training is carried out until convergence with early stopping to prevent overfitting. Trained artifacts (model weights, scaler parameters, metadata) are saved to a local registry for later loading during streaming inference.

F. Evaluation Metrics

Performance is quantified using standard forecasting metrics [29]–[32]: MAPE $= \frac{100}{N} \sum_{i=1}^{N} \left| \frac{p_i - \hat{p}_i}{p_i} \right|$, RMSE $= \sqrt{\frac{1}{N} \sum_{i=1}^{N} (p_i - \hat{p}_i)^2}$, MAE $= \frac{1}{N} \sum_{i=1}^{N} |p_i - \hat{p}_i|$.

In addition to predictive accuracy, end-to-end latency (producer $\rightarrow$ consumer $\rightarrow$ evaluator) is recorded. These measurements capture both algorithmic efficiency and system-level responsiveness.

G. Baselines

For fairness, baseline forecasters, namely LSTM and TS-Mixer, are implemented under identical preprocessing, training, and evaluation protocols. Their inclusion enables a controlled comparison, but the central analysis emphasizes the advantages of KAN in terms of accuracy and suitability for edge streaming deployment.

IV. Experimental Setup

The code, configuration files, and processed datasets used in this study would be shared directly upon reasonable request from: https://github.com/anhdatle/.

A. Local Deployment Feasibility

Experiments were conducted on a personal laptop equipped with an Apple M1 CPU (8 cores, 16 GB unified memory, 512 GB SSD storage), running macOS with Docker containers for Kafka services. Kafka provided a lightweight message bus for producing asset ticks and consuming predictions in real time. Each run instantiated a fresh topic to ensure isolation and reproducibility. The entire pipeline, including data ingestion, preprocessing, model training, and streaming inference, was executed without access to GPUs or external cloud resources. Models (KAN, LSTM, TS-Mixer) were implemented in PyTorch, with training carried out on the CPU. Each trained model occupied less than 20 MB on disk, and the Kafka broker persisted topics locally with storage overhead below 200 MB per asset stream. This confirms that the architecture is lightweight and can be reproduced on commodity hardware, making it suitable for edge deployments where compute and memory budgets are constrained. Unlike prior work that often depends on distributed clusters or cloud-based pipelines, the present design validates the feasibility of accurate and efficient forecasting on a single personal machine.

B. Data Sources

Historical daily close prices were retrieved using `yfinance` [33] with fallbacks to Stooq and FRED in case of missing data. Five assets were collected:

- **WTI Crude Oil (CL=F)** – futures contract for West Texas Intermediate crude oil.
- **Gold (GC=F)** – gold futures contract.
- **Silver (SI=F)** – silver futures contract.
- **Copper (HG=F)** – futures contract for high-grade copper.

For the main evaluation, four assets (WTI, gold, silver, copper) were highlighted. Each dataset spans January 2018 to Sep 2025, yielding 1900 trading days per asset. After scaling with MinMax, sequences were prepared using a sliding window of 16 days and a one–day prediction horizon.

C. Models

The study focused on KAN as the proposed forecaster. Baseline models included LSTM and TS-Mixer All models were implemented in PyTorch and trained under identical preprocessing conditions. Training used Adam optimizer with learning rates tuned per model class (KAN: 10^{-2}, LSTM: 10^{-3}, TS-Mixer: $2x10^{-3}$). The learning rates differ across models because each architecture has distinct gradient dynamics and stability requirements, requiring tuning to ensure both fast and stable convergence. Each model was trained for 100 steps with early stopping, splitting 80% of the series for training and 20% for testing.

D. Streaming Protocol

After offline training, fitted models were exported into a local registry (`/models`) with artifacts (`model.pt`, scaler, manifest). Online evaluation used a Kafka-based producer to replay test windows as ticks and a consumer to perform predictions. Each prediction was joined with the next observed tick to compute accuracy per step. End-to-end latency was measured from tick ingestion to next-tick validation, decomposed into inference time and system overhead.

E. Evaluation Metrics

Three accuracy metrics were reported: Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Latency was reported by end-to-end latency per tick (ingest $\rightarrow$ predict $\rightarrow$ validation) [29], [30].

V. Results and Analysis

This section presents the empirical evaluation of forecasting accuracy and latency across four commodity assets (WTI crude oil, gold, silver, and copper). Accuracy is measured using Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Latency is reported by end-to-end (E2E) streaming latency per tick, including data transport and coordination overhead. The best results for each metric are highlighted in **bold**.

A. Accuracy Analysis

1) WTI Crude Oil: As shown in Table I, KAN achieved the best performance with a MAPE of 1.83%, RMSE of 1.60, and MAE of 1.41. This significantly outperformed TS-Mixer (MAPE = 3.81%) and LSTM (MAPE = 6.89%), demonstrating that KAN captures oil price dynamics with high precision and reduces forecasting error by more than half compared to LSTM.

TABLE I
Forecasting accuracy on WTI crude oil (test set).

Model	MAPE (%)	RMSE	MAE
KAN	**1.83**	**1.60**	**1.41**
TSMixer	3.81	3.29	2.98
LSTM	6.89	5.73	5.38

2) Gold: Table II shows that for gold, KAN achieved the best performance across all three metrics, with a MAPE of 2.36%, RMSE of 69.64, and MAE of 50.07. This is substantially better than LSTM, which recorded a high error (MAPE = 15.84%), and also outperformed TS-Mixer, which achieved slightly higher errors (MAPE = 2.66%, RMSE =

70.85, MAE = 56.08). These results highlight that KAN is especially effective in forecasting gold prices, even within the volatile precious metals category.

TABLE II
FORECASTING ACCURACY ON GOLD (TEST SET).

Model	MAPE (%)	RMSE	MAE
KAN	**2.36**	**69.64**	**50.07**
TSMixer	2.66	70.85	56.08
LSTM	15.84	333.68	328.84

3) Silver: The results for silver are presented in Table III. KAN outperformed both baselines with a MAPE of 2.17%, RMSE of 0.62, and MAE of 0.51. TS-Mixer followed with 3.08% error, while LSTM lagged significantly at 8.13%. These results demonstrate KAN's robustness for forecasting moderately volatile assets such as silver.

TABLE III
FORECASTING ACCURACY ON SILVER (TEST SET).

Model	MAPE (%)	RMSE	MAE
KAN	**2.17**	**0.62**	**0.51**
TSMixer	3.08	0.91	0.73
LSTM	8.13	2.05	1.91

4) Copper: Table IV highlights that KAN delivered the best performance on copper with a MAPE of 1.54%, RMSE of 0.07, and MAE of 0.06. TS-Mixer showed competitive results (MAPE = 1.74%), whereas LSTM underperformed with 5.73%. This underscores KAN's effectiveness in modeling base metals and achieving reliable accuracy across datasets.

TABLE IV
FORECASTING ACCURACY ON COPPER (TEST SET).

Model	MAPE (%)	RMSE	MAE
KAN	**1.54**	**0.07**	**0.06**
TSMixer	1.74	0.08	0.07
LSTM	5.73	0.24	0.22

Across all four commodities, KAN consistently outperformed LSTM and achieved competitive or superior results compared to TS-Mixer. KAN was strongest for WTI, silver, and copper, while TS-Mixer was slightly better on gold. These findings demonstrate that KAN is a strong, general-purpose forecaster for streaming commodity prices, especially effective for energy and base metals, while complementary models may retain advantages in precious metals.

5) End-to-End Latency.: Table V reports the end-to-end latency per tick, which includes the full Kafka streaming pipeline (data ingestion, prediction, and next-tick validation). Across all assets, the average latency is around 1.14–1.17 seconds, with median (p50) values close to 1.21–1.23 seconds and tail (p95) delays between 1.79 and 1.83 seconds. The results are consistent across WTI, gold, silver, and copper, showing that networking and coordination dominate the total latency, while the choice of forecasting model contributes only marginally. This confirms that the system remains stable and predictable in a streaming setting.

TABLE V
END-TO-END LATENCY PER TICK INCLUDING KAFKA STREAMING OVERHEAD.

Asset	Avg (ms)	p50 (ms)	p95 (ms)
WTI	1142.9	1231.8	1817.6
Gold	1146.8	1213.9	1807.3
Silver	1168.8	1236.8	1830.6
Copper	1165.0	1234.4	1794.7

VI. LIMITATIONS

A. Market and data scope.

The study uses end-of-day (EOD) settlement prices for major commodities. Intra-day dynamics (microstructure noise, liquidity shocks, cross-venue fragmentation) and transaction frictions (bid–ask spreads, slippage, funding costs) are not modeled. Futures contract specifics (roll conventions, holiday calendars, and exchange trading halts) can introduce small alignment errors across sources even after preprocessing. Data gaps and late updates were mitigated with fallbacks (Stooq, FRED) and forward-filling only within trading days, but residual mismatches may remain.

B. Modeling assumptions.

All forecasters operate on univariate sliding windows with a one-step horizon and MinMax scaling fitted on the training split only. This deliberately avoids leakage but does not capture cross-asset dependencies, regime switches, or structural breaks explicitly. Hyperparameters were kept compact to reflect an edge-oriented design rather than global optimum accuracy; more aggressive tuning could further improve baselines or KAN. The evaluation reports point forecasts only; uncertainty quantification (prediction intervals) and risk-aware decision metrics are out of scope.

C. System considerations.

Latency measurements reflect a single-node Docker/Kafka setup on commodity CPU hardware. Results may vary with different brokers (e.g., replication factor, retention policies), OS scheduling, container runtime, and background I/O.

VII. CONCLUSION

This paper presented a compact, edge-oriented streaming forecaster centered on Kolmogorov–Arnold Networks (KAN) and embedded in a realistic Kafka loop. The design emphasizes small models, minimal infrastructure, and a reproducible next-tick evaluation protocol. Across four commodities (WTI, gold, silver, copper), KAN consistently delivered strong point-forecast accuracy, surpassing LSTM and matching or exceeding TS-Mixer on three of four assets in the reported runs.

A key systems result is that user-visible delay is dominated by streaming overheads—serialization, broker coordination, and consumer polling—rather than model compute. Inference on CPU for all models contributed only a minor fraction of the end-to-end path, suggesting that, for many practical edge deployments, engineering attention should prioritize topic design, batching/linger tuning, and consumer fetch policies before further shrinking already-small models.

The combination of (i) a lightweight forecaster, (ii) a transparent data and preprocessing pipeline with public fallbacks, and (iii) an online, per-tick measurement harness provides a practical template for network-aware ML at the edge. Future work includes multi-horizon and probabilistic forecasting, explicit regime/draft detection, cross-asset conditioning, quantization and CPU vectorization for additional efficiency, as well as broader sensitivity studies on broker parameters (replication factor, retention, fetch/linger) to map system knobs to latency/throughput trade-offs.

REFERENCES

[1] Walter C Labys and Peter K Pollak. *Commodity models for forecasting and policy analysis*. Routledge, 2024.

[2] John Baffes and Peter Nagle. *Commodity markets: evolution, challenges, and policies*. World Bank Publications, 2022.

[3] Lei Ge, Qiwei Huang, Fengshuang Zhu, and Shun Chen. Advanced time series forecasting for commodities: Insights from the fedformer model. *Energy Economics*, page 108513, 2025.

[4] Harshayu Girase, Nakul Agarwal, Chiho Choi, and Karttikeya Mangalam. Latency matters: Real-time action forecasting transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18759–18769, 2023.

[5] Dozdar Mahdi Ahmed, Masoud Muhammed Hassan, and Ramadhan J Mstafa. A review on deep sequential models for forecasting time series data. *Applied computational intelligence and soft computing*, 2022(1):6596397, 2022.

[6] Bill Bejeck. *Kafka Streams in Action: Event-driven Applications and Microservices*. Simon and Schuster, 2024.

[7] Iynkaran Natgunanathan, Vicky Mak-Hau, Sutharshan Rajasegarar, and Adnan Anwar. Deakin microgrid digital twin and analysis of ai models for power generation prediction. *Energy Conversion and Management: X*, 18:100370, 2023.

[8] Laura Melgar-García, David Gutiérrez-Avilés, Cristina Rubio-Escudero, and Alicia Troncoso. A novel distributed forecasting method based on information fusion and incremental learning for streaming time series. *Information Fusion*, 95:163–173, 2023.

[9] Tobias Meuser, Lauri Lovén, Monowar Bhuyan, Shishir G Patil, Schahram Dustdar, Atakan Aral, Suzan Bayhan, Christian Becker, Eyal De Lara, Aaron Yi Ding, et al. Revisiting edge ai: Opportunities and challenges. *IEEE Internet Computing*, 28(4):49–59, 2024.

[10] Lixin Yan, Le Jia, Shan Lu, Liqun Peng, and Yi He. Lstm-based deep learning framework for adaptive identifying eco-driving on intelligent vehicle multivariate time-series data. *IET Intelligent Transport Systems*, 18(1):186–202, 2024.

[11] Margustin Salim and Arif Djunaidy. Development of a cnn-lstm approach with images as time-series data representation for predicting gold prices. *Procedia Computer Science*, 234:333–340, 2024.

[12] Javid Iqbal, Aneeza Ahmed, and Muhammad Ramzan. Forecasting the nexus and impact of news sentiment on nyse, gold prices, and wti oil using the neural network approach. *Bahria University Journal Of Management & Technology*, 7(1), 2024.

[13] Dat Le, Sutharshan Rajasegarar, Wei Luo, Thanh Thi Nguyen, and Maia Angelova. Navigating uncertainty: Gold price forecasting with kolmogorov-arnold networks in volatile markets. In *2024 IEEE Conference on Engineering Informatics (ICEI)*, pages 1–9. IEEE, 2024.

[14] Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. Kan 2.0: Kolmogorov-arnold networks meet science. *arXiv preprint arXiv:2408.10205*, 2024.

[15] Dat Le, Sutharshan Rajasegarar, Wei Luo, Thanh Thi Nguyen, and Maia Angelova. Gold price forecasting in uncertain times: Integrating sentiment analysis and external indices. In *2024 11th International Conference on Soft Computing Machine Intelligence (ISCMI)*, pages 211–215, 2024.

[16] Dat Le, Sutharshan Rajasegarar, Wei Luo, Thanh Thi Nguyen, and Maia Angelova. Hybrid kolmogorov-arnold and graph attention networks for gold price forecasting under uncertainty. In Tianqing Zhu, Wanlei Zhou, and Congcong Zhu, editors, *Knowledge Science, Engineering and Management*, pages 443–454, Singapore, 2026. Springer Nature Singapore.

[17] Sami Ben Jabeur, Salma Mefteh-Wali, and Jean-Laurent Viviani. Forecasting gold price with the xgboost algorithm and shap interaction values. *Annals of Operations Research*, 334(1):679–699, 2024.

[18] Jiawei Shao and Jun Zhang. Communication-computation trade-off in resource-constrained edge inference. *IEEE Communications Magazine*, 58(12):20–26, 2021.

[19] Hamed Z Jahromi, Andrew Hines, and Declan T Delanev. Towards application-aware networking: Ml-based end-to-end application kpi/qoe metrics characterization in sdn. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 126–131. IEEE, 2018.

[20] J Efrim Boritz and Won Gyun No. How significant are the differences in financial data provided by key data sources? a comparison of xbrl, compustat, yahoo! finance, and google finance. *Journal of Information Systems*, 34(3):47–75, 2020.

[21] Stooq. Stooq financial data portal, 2025. Accessed: 2025-09-25.

[22] Federal Reserve Bank of St. Louis. Fred: Federal reserve economic data, 2025. Accessed: 2025-09-25.

[23] Baris Kocaarslan and Ugur Soytas. How do the reserve currency and uncertainties in major markets affect the uncertainty of oil prices over time? *International Journal of Finance & Economics*, 2024.

[24] Guanghao Wang, Chenghao Liu, Erwann Sbai, Mingyue Selena Sheng, Jinhong Hu, and Miaomiao Tao. Interrelations between bitcoin market sentiment, crude oil, gold, and the stock market with bitcoin prices: Vision from the hedging market. *Studies in Economics and Finance*, 2024.

[25] Srilekha Nallamothu, K Rajyalakshmi, and P Arumugam. Gold price prediction using skewness and kurtosis based generalized auto-regressive conditional heteroskedasticity approach with long short term memory network. *Journal of The Institution of Engineers (India): Series B*, pages 1–13, 2024.

[26] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[27] Docker, Inc. Docker: Accelerated, containerized application development. https://www.docker.com/, 2025. Accessed: 2025-09-25.

[28] Apache Software Foundation. Apache kafka: A distributed streaming platform. https://kafka.apache.org/, 2025. Accessed: 2025-09-25.

[29] Rob Hyndman and Anne Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688, 2006.

[30] Francis Diebold and Roberto Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, 1995.

[31] Tilmann Gneiting and Adrian Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378, 03 2007.

[32] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[33] Ran Aroussi. yfinance: Yahoo! finance market data downloader. https://github.com/ranaroussi/yfinance, 2018. Accessed: 2025-09-26.