Final Report COMP 433

Youssef Ouakaa (40157718)
Gabriel Asencios (40176253)
Nicolae Rusu (40245233)
Ryan Mazari (40241379)

Abstract

Accurate time-series forecasting is essential in domains such as climate science, energy management, economics, and e-commerce, where decisions depend on anticipating future trends from historical observations. This report focuses on the Retrieval-Augmented Time-Series Forecasting (RAFT) framework, which integrates a shallow MLP forecaster with a retrieval module that selects similar historical segments from the entire time series. We reproduce the experiments from the original RAFT paper across benchmark datasets spanning energy, transportation, finance, health, and weather, and evaluate performance using Mean Squared Error (MSE) and Mean Absolute Error (MAE). We then extend the evaluation to an e-commerce sales dataset to assess generalization in a highly non-stationary business setting.

1. Introduction

1.1. Problem Statement

Accurate time-series forecasting is essential across numerous scientific and industrial domains, especially in time-dependent applications where decisions need to be made by anticipating future trends based on historical observations. Such applications include climate science, energy management, economics, and user behavior analysis [1, 5, 9, 25]. E-commerce is one such domain where forecasting has become increasingly vital, yet the rapid adoption of online retail has introduced unique challenges characterized by complex, non-stationary sales patterns that shift unpredictably due to promotions, seasonality, and external market factors. Traditional forecasting methods struggle with these dynamics, while modern deep learning approaches that can capture this complexity often demand substantial computational resources.

More broadly, real-world time series present difficulties for existing forecasting models across all domains due to complex, non-stationary patterns that vary in peaks, periods, and shapes. In e-commerce sales, these patterns are explained by sales, holidays, and recommendation algorithms; in energy applications, it's mostly explained by weather, etc. Since these patterns might lack time correlation and might be generated by non-deterministic causes, they result in infrequent repetitions and various statistical distributions. This causes problems for the existing prediction models in extrapolating from these irregular patterns, and when all patterns are learned and memorized by the model, including noisy and random ones, accuracy and generalizability are affected.

Traditionally, statistical methods such as ARIMA and exponential smoothing have formed the basis for time-series forecasting, but the field has naturally shifted towards deep learning approaches. Early approaches used CNNs to identify local temporal patterns [3] and RNNs with LSTM variants to detect patterns from past data [11]. More recently, the field has moved towards the direction of Transformer-based architectures, which showed strong results and captured complex input patterns. Examples include Autoformer [21] and FEDformer [24], which enhanced attention mechanisms to identify trends.

However, these transformer-based models, despite being effective, require substantial computational resources, making them difficult to utilize for real-time industrial applications. Moreover, studies have demonstrated that lightweight models can achieve high performance by decomposing the time series and conducting a multi-periodicity analysis.

To address these limitations, lightweight retrieval augmented architectures appeared as a promising research direction. The RAFT framework proposed by the paper “Retrieval Augmented Time Series Forecasting” [10] integrates a simple shallow MLP module for forecasting with a retrieval mechanism that uses similar historical segments, achieving strong performances without the computational resources or architectural complexity of Transformers. The method is inspired by the popular retrieval-augmented generation (RAG) often used in large language models [14].

This approach offers two main advantages that address the issues mentioned above. First, by using retrieved historical segments explicitly at inference time, learning can cover patterns that do not have any temporal correlation in the series. Second, the retrieval mechanism allows the model to leverage rare historical patterns when they reappear.

In this report, we reproduce the results of the RAFT paper [10] and validate their claims by conducting the experiments across the benchmark datasets cited in the paper. We expect to achieve the performances cited in the paper to demonstrate the effectiveness of retrieval augmented forecasting in comparison to state-of-the-art baseline transformer models, with a high computational efficiency. For these benchmarks, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as metrics. Building upon this, once the basic results are reproduced, we extend the testing of the model to an e-commerce sales dataset to evaluate generalization. For all of these, we visualize results with time-series plots comparing predicted and actual trajectories.

1.2. Related Works

Time-series forecasting has significantly progressed over the past decade, especially with the advances in deep learning methods. Models such as CNNs [3] and RNNs [11] were used at first, achieving effective results in capturing patterns in historical data. These early deep learning methods laid the foundation for more advanced architectures to be later used for time-series forecasting.

Significant improvements were made with the advent of attention mechanisms and transformer-based architectures [19]. These models were effective in capturing the dependencies and intricacies in time-based data, resulting in several proposed models like AutoFormer [21] and FedFormer [24]. AutoFormer and FedFormer both decompose the series into components to have a more accurate prediction by identifying trends and seasonal patterns. These decomposition techniques help the models learn characteristics with different temporal behaviors.

In addition to the new transformer architectures, recent methods are using techniques like time series decomposition and multi-periodicity analysis involving downsampling and upsampling the series at various period intervals [20].

Despite the advancements in transformer-based models, lightweight models for time series forecasting based on MLPs have achieved good performances when complemented with other techniques. Chen et al. [6] showed that even a simple MLP-based approach can achieve competitive performances for forecasting, questioning the necessity of very complex and resource-consuming architectures. This discovery led to the appearance of several lightweight MLP-based models such as TiDE [7], TSMixer [6], and TimeMixer [20] that were developed to address training efficiency in time-series forecasting. These methods implement many approaches such as series decomposition and multi-periodicity mentioned above, to extract relevant information for the MLPs to learn from or use at inference time.

Beyond time-series forecasting, we look at the other aspect presented by the paper which is retrieval augmented generation (RAG) which has gained popularity, mostly in natural language processing and large language models [14]. Traditionally, RAG retrieves document chunks relevant to the task at hand, from external corpuses to help the LLM response to be relevant and to prevent hallucination [2, 14]. This shows that supplementing the model’s inputs with retrieved information can be more efficient than encoding all knowledge in the model’s weights solely.

Retrieval-augmented techniques have been applied to other structured data problems outside of natural language processing. Some approaches have applied attention-based retrieval on tabular and structured data [8].

There also exists work exploring the potential of retrieving similar segments in time-series forecasting before the RAFT paper was introduced [12, 22]. However, these methods focused on multiple time-series and meta learning which is different from the focus of the RAFT paper, where there is only one time series available for training.

Similar to how RAG is used in LLMs to supplement the input with additional information, RAFT aims to reduce the learning complexity in time-series by supplementing the input with the needed context. The MLP model does not have to learn the complex patterns through its weights which are kept simple. An additional retrieval model is appended and simplifies the learning process. This also improves the performance for datasets with spikes and irregularities.

2. Methodology

In this section, we describe the implementation of the Retrieval-Augmented Forecasting of Time-Series (RAFT) proposed by the paper [10]. We start with an overview of the whole architecture, followed by a detailed explanation of the retrieval mechanism and the forecast module as they were presented in the paper, and finally, a small overview of the datasets.

2.1. Architecture Overview

The RAFT consists of two main modules. The retrieval module to find relevant patches to the query used as an input, and an MLP for the forecasting that is fed the retrieved patches as input and predicts future values y. An overview of this architecture is illustrated in Figure 1.

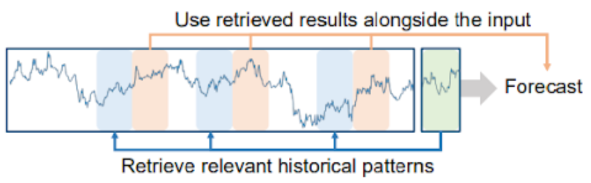
Use retrieved results alongside the input
Forecast
Retrieve relevant historical patterns

Figure 1. High-level illustration of the RAFT method [10].

Given a time series $S \in \mathbb{R}^{C\times T}$ of length T with C variates or channels (columns in the Python implementation), RAFT uses historical observation $x \in \mathbb{R}^{C\times L}$ to predict future values $y \in \mathbb{R}^{C\times F}$ that are as close as possible to the actual future values y_0. In this case, hyperparameters L and F denote the look-back window size and the forecasting window size, respectively.

The key idea proposed by the paper is to augment the forecasting module with a retrieval mechanism that provides similar historical patterns from the entire time series before passing them to the forecasting module, unlike traditional attention-based models that have a fixed lookback window.

Given an input x, as described above, the retrieval module finds the m most relevant patches, where m is a hyperparameter to the model. For each retrieved patch, the subsequent patches are also retrieved, providing additional information about the context. Importance weights are calculated based on the correlation between the input x and the patches, and they are aggregated through a weighted sum, similar to how the attention mechanism works in other popular architectures. The main difference is that the RAFT retrieval model can retrieve data from the entire time series and not from a single fixed lookback window. This mechanism allows the model to detect patterns in arbitrary periods in the data, as long as they are similar or correlated to the input.

Another addition is that the retrieval module can aggregate results with multiple different periods, since trends and temporal patterns might have different time scales. This is done by downsampling the time series with different periods P and applying the retrieval module described above to each time series resulting from the downsampling. The retrieved results are processed by a linear projection and summed. The final result is concatenated with the inputs and passed to the linear MLP for producing the final prediction.

2.2. Retrieval Module

In this section, we describe in more detail the retrieval mechanism both for single periods and multiple periods. The retrieval mechanism works by extracting key-value pairs based on an input x. First, the time-series S is split into a collection

$$K = \{k_1, \dots, k_{T-(L+F)+1}\}$$

of key patches with a sliding window of length L with a stride of 1. The indices indicate the starting time step of the patches, knowing they have length L, hence $k_i \in \mathbb{R}^{C\times L}$. Any patch overlapping with the given input x is excluded from K during training to ensure a better generalization.

Using the key patches defined above, a collection of value patches is defined, of length F similar to the expected prediction,

$$V = \{v_1, \dots, v_{T-(L+F)+1}\},$$

where each $v_i \in \mathbb{R}^{C\times F}$ sequentially follows after k_i in the time series.

After creating the key patch set K and the value set V, the input x is used as a query to retrieve the most similar keys and their corresponding value patches. For all patches, the numerical value at the final time step is considered an offset and subtracted from all elements in the patch as a form of preprocessing, giving us $\hat{x}$, $\hat{K}$ and $\hat{V}$. This is inspired by existing literature and acts as some form of normalization.

Then, the Pearson correlation is used as a similarity function to find the similarity ρ_i between given $\hat{x}$ and all key patches in $\hat{K}$:

$$\rho_i = s(\hat{x}, \hat{k}_i), \quad \hat{k}_i \in \hat{K}.$$

The patches with top-m correlations are retrieved, knowing m is a hyperparameter, and obtaining J representing the indices of the top-m patches.

Using another hyperparameter, temperature τ, value patches are assigned weights with the following equation:

$$w_i = \begin{cases} \dfrac{\exp(\rho_i/\tau)}{\sum_{j\in J}\exp(\rho_j/\tau)}, & i \in J, \\ 0, & \text{otherwise.} \end{cases}$$

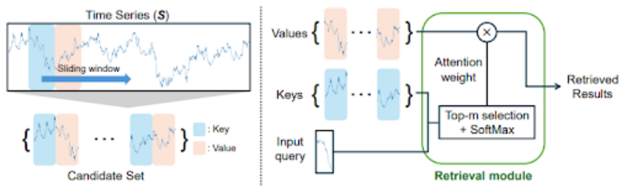

Figure 2. Retrieval mechanism [10].

The final retrieval result is simply the weighted sum of value patches:

$$\tilde{v} = \sum_{i=1}^{|V|} w_i\, v_i.$$

This weighted sum ensures that the most similar historical patterns have the most impact on the prediction while still allowing other less relevant patches to provide some information, as summarized in Figure 2.

2.3. Forecast Module

Single period. Given the input x and the retrieved patch $\tilde{v}$, the offset is subtracted from x to obtain $\hat{x}$ similarly to what was done in the retrieval process, and then the model concatenates $f(\hat{x})$ and $g(\tilde{v})$ where f and g are both linear projections to $\mathbb{R}^F$ and the result is processed to obtain an output in $\mathbb{R}^F$. The final expression for the output $\hat{y}$ is:

$$\hat{y} = h\big(f(\hat{x}) \oplus g(\tilde{v})\big)$$

representing what is done by the linear MLP.

Multiple periods. Multiple periods are used to retrieve both local patterns from small time windows and global trends from larger time windows. We consider n periods P. For each period, the query x is downsampled with average pooling and we obtain $x^{(p)}$, $K^{(p)}$, and $V^{(p)}$, and $\tilde{v}^{(p)}$ for each p. Then for all values of p, the retrieval results are processed with a linear layer $g^{(p)}$, projected to the same dimensions, summed and concatenated with the linear projection of the input. The result is passed to a linear layer h to obtain the

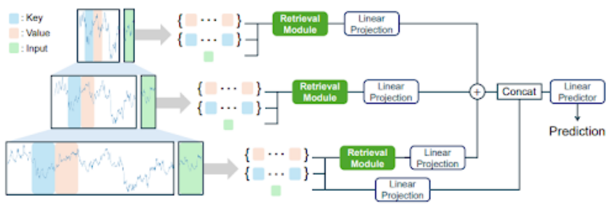

Figure 3. RAFT with multiple periods [10].

final result $\hat{y}$ given by the formula:

$$\hat{y} = h\left(f(\hat{x}) \oplus \sum_{p\in P} g^{(p)}\left(\tilde{v}^{(p)}\right)\right).$$

The overall multi-period RAFT architecture is illustrated in Figure 3.

For both approaches, after finding $\hat{y}$, the offset x_L is added back to every time step in $\hat{y}$ to obtain the final forecast y and the model is trained by minimizing:

$$\mathcal{L} = \text{MSE}(y, y_0).$$

2.4. Datasets

Benchmark datasets from the RAFT paper. RAFT was evaluated on ten public datasets spanning energy, economics, health, and transportation. All datasets are multivariate, with train/validation/test splits and normalized features. For training, sliding-window sequences with look-back L and forecast horizon F are generated. The datasets include:

- **ETTh1 and ETTh2 (Energy, hourly):** Transformer temperature measurements (2016–2018) [23].
- **ETTm1 and ETTm2 (Energy, 15 min):** Same source at higher frequency [23].
- **Electricity (Energy, hourly):** Household power consumption over 4 years [18].
- **Exchange (Finance, daily):** Exchange rates of 8 countries (1990–2016) [13].
- **Illness (Health, weekly):** U.S. influenza-like illness ratios (2002–2021) [4].
- **Solar (Energy, 10 min):** Solar power production from plants in Alabama (2006) [16].
- **Traffic (Transportation, hourly):** Freeway occupancy rates from loop detectors [13].
- **Weather (Environment, 10 min):** Meteorological observations in Germany, including temperature, CO_2, and humidity [15].

Additional dataset (E-commerce sales). After validating the benchmarks, we evaluate an open e-commerce sales dataset due to its relevance in this context [17]. Retrieval can be powerful for non-stationary business data where external events influence trends. This dataset contains transactional sales data over time, which we convert into a multivariate time series with temporal and seasonal features to improve model accuracy.

3. Results

This section details in depth the experimental design used to optimize the strategy, validate the methodology, and the choice made in regards to hyperparameters across all datasets.

3.1. Experimental Setup and Data Processing

All experiments are conducted under the forecasting formulation proposed by the paper. In fact, for each dataset, a sliding window, a technique used for object detection and time series forecasting, are built to generate supervised training sequences. The ETTh1, ETTh2, ETTm1, and ETTm2 are all 15 minutes and hourly dataset that use a 96-step input window and 48-step look-ahead label segment. These datasets also use a 96-step forecasting horizon which describes the future time period over which predictions are made. On the other hand, datasets having lower frequencies like the E-commerce Sales (daily) use shorter windows. All datasets are normalized independently within each split. The RAFT dataloader automatically splits each dataset into predetermined train, validation, and test suites. For instance, the ETTh1 and ETTh2 are both allocated 8449, 2785, and 2785 samples respectively. On the contrary, the ETTm1 and ETTm2 which are of high resolution contain over 34,000 training test suites. Before training, the RAFT model performs a retrievalindex construction over the splits in order to allow the model to access historical partners during the learning phase.

3.2. Model Optimization and Validation Strategy

In order to optimize the models, we decide to adopt a consistent training protocol across the datasets. The RAFT model is trained using Adam optimizer with an initial learning rate of 1×10^{-4} and a dynamic decay schedule that has a role to reduce in half the learning rate whenever the validation loss does not move. As an example, in the ETTh2 dataset the learning rate is updated from 5×10^{-5} to 2.5×10^{-5} from epoch 2 to epoch 3. It is important to note that all models are trained for 10 epochs with a batch size of 32. Also, validation is performed on every epoch. For each epoch, the RAFT model reports the training, validation, and test losses. We use these validation metrics to guide the learning rate and adjust it. This can also help to find some early indicators of overfitting too. For example, in the ETTh1 validation loss stabilizes around 0.70 while the training loss is continuously decreasing. This indicates the model is not overfitting despite continuous optimization.

3.3. Performance Metrics and Evaluation Methodology

Since long-term time-series forecasting is a regression problem, the traditional metrics used in deep learning such as precision, F1, and recall are not applicable in this case. However, we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) to evaluate the model performance. First, the training validation loss is monitored at every epoch to detect overfitting. Then, epoch-level convergence patterns are analyzed in order to check for stability under the current learning rate. Lastly, the performance is quantified using the MSE and MAE at the final epoch. For example, the RAFT model achieved a test MSE of 0.4040 on ETTh1 and 0.2980 on the ETTh2. These metrics will be presented in the Main Results section and used for the comparative analysis.

3.4. Hyperparameter Rationale and Model Configuration

All experiments use a consistent set of hypermarkets in order to ensure a fair and controlled comparison across all datasets. The transformer uses a model dimension of 512, a feed-forward dimension of 2048, a two-layer encoder, height attention heads, and a single layer decoder. Also, the retrieval module uses Top-k=5 nearest neighbors and six convolutional kernels. We observed that having a k beyond five resulted in diminishing performances for most datasets. The de-stationary projection module is made of two hidden layers of size 128. This module is responsible for modulating input sequences in order to stabilize learning. Training hyperparameters like an initial learning rate of 1×10^{-4}, a batch size of 32, and 10 epochs were selected after yielding the best results based upon the paper. Based on these choices, we agree that they have been well chosen since even on small-scale datasets like the E-Commerce with only 464 training samples, the model still achieves satisfactory performance.

3.5. Main Results

Our reproduction of the RAFT framework confirms its efficacy and robustness under significant hardware constraints. For the ETTh1, ETTh2, ETTm1, and ETTm2 datasets [23], we replicated the performance reported by Han et al. [10] within a negligible margin (MSE $\Delta < 0.02$, see Table 1), despite limiting the lookback window to 48 steps due to memory constraints. On the Exchange dataset [13], this constrained configuration outperformed the original results [10], suggesting that shorter historical context can effectively regularize financial time-series predictions by filtering out distant noise.

However, experiments revealed critical trade-offs between model dimensionality and memory constraints on datasets with high feature counts. On the Solar dataset [13], 137 independent variates increased the retrieval cache's memory footprint. Restricting the lookback to 48 steps failed to capture the necessary 24-hour seasonal cycle, resulting in degraded performance (Table 1).

Table 1. Comparison between our reproduced RAFT results and the original paper on key datasets.

Dataset	Metric	Ours	Original	Δ	Status
ETTh1	MSE	0.387	0.367	+0.020	Comparable
ETTh1	MAE	0.414	0.397	+0.017	
ETTh2	MSE	0.296	0.276	+0.020	Comparable
ETTh2	MAE	0.350	0.344	+0.006	
ETTm1	MSE	0.329	0.302	+0.027	Comparable
ETTm1	MAE	0.371	0.349	+0.022	
ETTm2	MSE	0.177	0.164	+0.013	Successful
ETTm2	MAE	0.266	0.256	+0.010	
Weather	MSE	0.235 (L=720)	0.165	+0.070	Acceptable
Weather	MAE	0.278 (L=720)	0.222	+0.056	
Exchange	MSE	0.084	0.091	−0.007	Superior
Exchange	MAE	0.200	0.209	−0.009	
Solar*	MSE	0.216 (L=48)	0.192	+0.024	Degraded
Solar*	MAE	0.263 (L=48)	0.251	+0.012	
E-commerce	MSE	1.904 (L=24)	N/A	N/A	New Benchmark
E-commerce	MAE	1.085 (L=24)	N/A	N/A	

* Solar dataset used batch size 16.

The Weather dataset [15] indicates a correlation between context length and accuracy; increasing the lookback window from 48 to 720 improved MSE from 0.263 to 0.235 (Table 1). The remaining gap compared to the original baseline reported by Han et al. [10] of 0.165 might stems from two factors: GPU memory constraints that forced single-period retrieval ($n_{\text{period}} = 1$) rather than multi-period parallel training ($n = 3$) needed to capture global trends, and the inability to replicate the exact dataset version since the original study does not specify which year was used for their Weather dataset experiments, potentially introducing distribution shifts that contribute to this disparity (Table 1).

The plots from ETTh1 and Weather validate the differential impact of context length (L) and inherent architectural constraints on Retrieval Augmented Forecasting for Time series.

On the ETTh1 dataset (Figure 4), the model shows strong stability and reliability. Despite being restricted to a small lookback window of L=48, the prediction (orange line) correctly forecasts the complex hourly cycles and major directional shifts of the temperature signal. While the primary visual characteristic is a reduction in peak size (amplitude attenuation), this phenomenon, a common side effect of the retrieval mechanism, smoothing out past data, still successfully maintains a low Mean Absolute Error (MAE) [10].

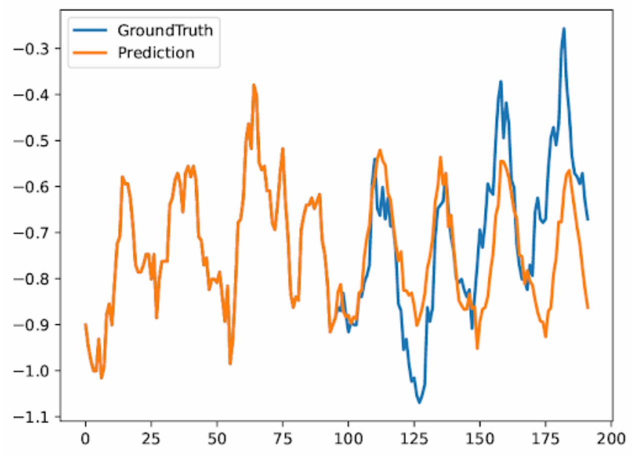

Figure 4. ETTh1 prediction visualization (Plot A).

In contrast, the Weather dataset (Figure 5), which required a full lookback (L=720) to achieve an acceptable result, still reveals a performance gap. Although the resulting Mean Squared Error (MSE 0.235) is highly competitive and often surpasses comparable

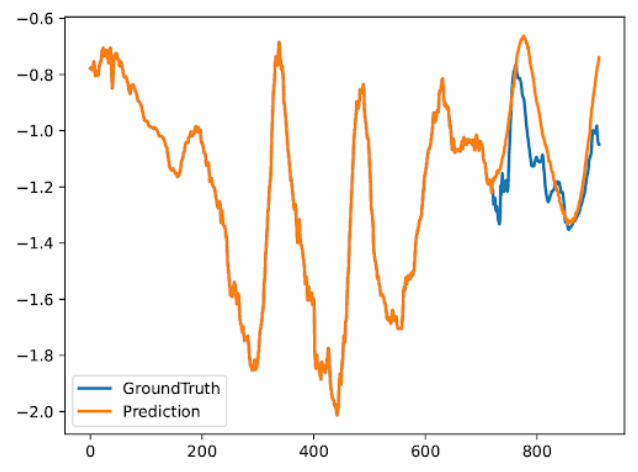

Figure 5. Weather prediction visualization (Plot B).

deep learning models tested in the original work [10], the remaining disparity confirms significant areas for improvement. While the prediction successfully captures the overall long-term trend and directionality, it exhibits predicted peaks that are too sharp (amplitude overestimation) and a noticeable timing delay (phase variation) in the recovery periods. The magnitude of this disparity confirms the combined problem of design limits and dataset disparity. We suspect the primary reason for the remaining performance gap lies in the data distribution shift resulting from the use of a different dataset than the one benchmarked in the original paper [10].

References

[1] Fabricio Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgilio Almeida. Characterizing user behavior in online social networks. In *IMC*, pages 49–62, 2009. 1

[2] Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens. *Proceedings of the 39th International Conference on Machine Learning*, 2022. 2

[3] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017. 1, 2

[4] Centers for Disease Control and Prevention (CDC). Fluview: Influenza-like illness surveillance. U.S. Department of Health and Human Services, 2021. https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html. 5

[5] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: Mining loop detector data. *Transportation Research Record*, 1748(1):96–102, 2001. 1

[6] Shang-An Chen, Chengrun Li, Sercan Ö. Arik, Nathan C. Yoder, and Tomas Pfister. TSMixer: An all-mlp architecture for time series forecasting. *Transactions on Machine Learning Research*, 2023. 2

[7] Abhimanyu Das, Weiyang Kong, Adam Leach, Suraj Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023. 2

[8] Yury Gorishniy et al. On feature interaction and retrieval in tabular deep learning. *arXiv preprint arXiv:2402.xxxxx*, 2024. 3

[9] Clive W. J. Granger and Paul Newbold. *Forecasting Economic Time Series*. Academic Press, 2 edition, 2014. 1

[10] Seungwoo Han, Seonghyeon Lee, Meeyoung Cha, Sercan Ö. Arik, and Jinsung Yoon. Retrieval-augmented time-series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, number 267 in PMLR, 2025. 2, 3, 4, 5, 6, 7, 8

[11] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time-series forecasting: Current status and future directions. *International Journal of Forecasting*, 2021. 1, 2

[12] Tomoharu Iwata and Atsutoshi Kumagai. Few-shot time-series forecasting via meta-learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 3

[13] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Multivariate time-series datasets for exchange, solar, and traffic benchmarks. GitHub repository, 2018. https://github.com/laiguokun/multivariate-time-series-data. 5, 6

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, pages 9459–9474, 2020. 2

[15] Max Planck Institute for Biogeochemistry. Weather dataset (germany). Department of Biogeochemical Processes, 2020. https://www.bgc-jena.mpg.de/wetter/. 5, 7

[16] National Renewable Energy Laboratory (NREL). Solar power data for integration studies. U.S. Department of Energy, 2020. https://www.nrel.gov/grid/solar-power-data. 5

[17] Prakash Rajak. E-commerce sales. Kaggle, 2024. https://www.kaggle.com/datasets/prince7489/e-commerce-sales. 5

[18] Alexandre Trindade. Electricity load diagrams 2011–2014. UCI Machine Learning Repository, 2015. https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014. 5

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30, 2017. 2

[20] Shiyang Wang, Haixu Wu, Xiaokang Shi, Tian Hu, Hongzhi Luo, Liang Ma, Jieyu Zhang, and Jing Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024. 2

[21] Haixu Wu, Jianmin Xu, Jingfei Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, pages 22419–22430, 2021. 1, 2

[22] J. Yang et al. Hierarchical time series forecasting with similarity-based retrieval. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. 3

[23] Haoyi Zhou. ETDataset: Electricity transformer temperature dataset (ett). GitHub repository, 2021. https://github.com/zhouhaoyi/ETDataset. 5, 6

[24] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR, 2022. 1, 2

[25] Ying Zhu and Dennis Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, pages 358–369, 2002. 1