[1]

Implicit Reasoning in Deep Time Series Forecasting

Willa Potosnak[1], Cristian Challu[1, 2], Mononito Goswami[1], Michał Wiliński[1], Nina Żukowska[1], Artur Dubrawski[1]
[1]Auton Lab, School of Computer Science, Carnegie Mellon University
[2]Nixtla

Abstract

Recently, time series foundation models have shown promising zero-shot forecasting performance on time series from a wide range of domains. However, it remains unclear whether their success stems from a true understanding of temporal dynamics or simply from memorizing the training data. While implicit reasoning in language models has been studied, similar evaluations for time series models have been largely unexplored. This work takes an initial step toward assessing the reasoning abilities of deep time series forecasting models. We find that certain linear, MLP-based, and patch-based Transformer models generalize effectively in systematically orchestrated out-of-distribution scenarios, suggesting underexplored reasoning capabilities beyond simple pattern memorization.

1 Introduction

Foundation models have demonstrated an exceptional ability to generalize to previously unseen data in zero-shot prediction tasks. Inspired by the success of such models in Natural Language Processing, recent work has adapted Transformers to build time series foundation models (TSFM). Zero-shot inference is particularly important for time series models, which must handle complex patterns, seasonal variations, and emerging trends where little to no reference data may be available.

Foundation models are trained on large, diverse datasets, raising a critical question for time series forecasting: **do these models generalize well because they learn underlying concepts of temporal dynamics, or do they simply memorize specific patterns seen during training?** If models rely on memorization, particularly in the form of time series pattern matching, it could lead to redundant knowledge storage, parameter inefficiency, and limit their ability to generalize well to out-of-distribution (OOD) data. Ideally, a TSFM should be capable of implicit reasoning, allowing it not to depend solely on memorization but to infer latent temporal dynamics. Such models would be able to generalize from fewer data points, offering enhanced parameter efficiency and robustness.

While extensive research has been conducted to evaluate memorization and implicit reasoning in language models, similar evaluations for time series models have been largely unexplored. In this work, we take an initial step toward evaluating the implicit reasoning capabilities of time series models in forecasting tasks. Our findings highlight the potential of linear, MLP-based, and patch-based Transformer models to perform well in carefully orchestrated OOD scenarios, suggesting that these models may have untapped capabilities in reasoning beyond mere memorization.

*Correspondence to: Willa Potosnak <wpotosna@andrew.cmu.edu>

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

2 Related work

Implicit Reasoning in LLMs. Prior research has explored implicit reasoning in language models [1, 2, 25, 30, 32]. Implicit reasoning is often assessed through tasks that require models to apply knowledge learned during training to new test instances. One common form is *composition*, where models must chain multiple facts to answer a question [25, 30, 32]. Other forms of implicit reasoning explored include *comparison* and *inverse search* [1, 25]. Comparison involves models evaluating two or more entities to make judgments, such as determining whether the attribute value of one entity is greater or smaller than that of another. Inverse search tests a model's ability to generate predictions in the reverse order of the training task. For example, this could involve applying the model to identify an entity based on its attributes when it was originally trained to predict the attributes of entities. More information on related work is provided in Appendix A.1. No prior research has conducted controlled experiments in time series forecasting to evaluate implicit reasoning on OOD data. Our study addresses this gap by introducing a novel framework that aligns LLM research with time series models, offering insights into optimal architectures for future TSFM development.

Time Series Foundation Models. Several foundation models, including `Chronos` [3], `LagLlama` [20], `Moirai` [27], `MOMENT` [11], `TimesFM` [7], `TimeGPT` [10], `Timer` [16], and `Tiny Time Mixers` [8], have been developed for time series forecasting. Some studies have also examined the impact of learning with synthetic data [3, 7]. `MOMENT` analyzed embeddings from synthetic sinusoids to isolate components like trends, while `Chronos` showed strong performance in forecasting composite series but struggled with exponential trends. These works demonstrate that TSFMs can learn distinct time series functions, though it remains unclear if their success is due to large-scale training or inherent reasoning abilities.

3 Methods

3.1 Implicit Reasoning Tasks

We assess the implicit reasoning capabilities of models, or their ability to apply and manipulate internalized patterns in novel situations, through experiments inspired by language model research [1, 2, 25, 30, 32]. We carefully design experiments to test 3 forms of implicit reasoning: *composition*, *comparison*, and *inverse search*. We follow the notation from [25, 32], where 'facts' are formatted as *(subject, relation, object)*. For continuous data, we adapt this to *(input, function, output)*, with $f(x) = y$ defining the relationship between the function f, input x, and output y. We consider two sets of basis functions: $\mathcal{F}_c = \{f : f(x) = mx \mid m \in \mathbb{R}\}$ and $\mathcal{F}_s = \{f : f(x) = a\sin(b2\pi x) + c \mid a, b, c \in \mathbb{R}\}$ where $\mathcal{F}_c$ is the set of all constant signals with linear trends m and $\mathcal{F}_s$ is the set of all sinusoidal signals with amplitude a, frequency b, and baseline shift c. We also define a set of functions consisting of compositions of basis functions: $\mathcal{F}_{sc} = \{f : f(x) = a\sin(b2\pi x) + c\,\{\pm, \times\}\, mx \mid a, b, c, m \in \mathbb{R}\}$. For a set of data-generating functions and their parameters included in the training set and referred to as in-distribution (ID) or $\mathcal{P}_{\text{train}}$, we assess a model's ability to generalize to other disjoint sets of data-generating functions observed only at inference, denoted as $\mathcal{P}_{\text{test}}$, which are considered OOD.

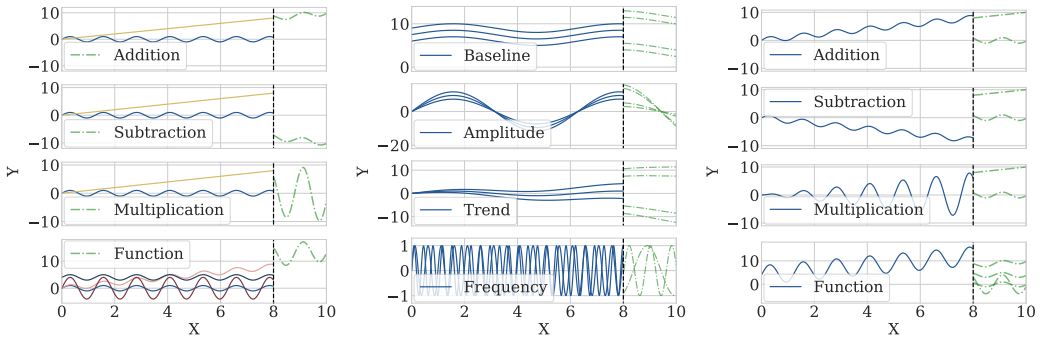
Composition. For models trained on signals generated from basis functions considered ID, we assess whether they can accurately forecast signal compositions that are considered OOD. Specifically, we define the datasets as follows: $\mathcal{P}_{\text{train}} = \mathcal{F}_s \cup \mathcal{F}_c$ and $\mathcal{P}_{\text{test}} = \mathcal{F}_{sc}$, where $\mathcal{P}_{\text{train}} \cap \mathcal{P}_{\text{test}} = \emptyset$. Consider a specific example:

$$f_s \in \mathcal{F}_s, f_c \in \mathcal{F}_c, f_{sc} \in \mathcal{F}_{sc}, \forall x \in \mathbb{R}, (x, f_s, y), (x, f_c, y) \in \mathcal{P}_{\text{train}} \text{ and } (x, f_{sc}, y) \in \mathcal{P}_{\text{test}} \quad (1)$$

Models are trained exclusively on the individual component series, and the compositions of series are withheld for final evaluations as illustrated in Fig. 1a. We train models on 30 trend and seasonality series ($n = 60$) and evaluate them on all compositions of these series ($n = 900$).

Comparison. Comparison refers to a model's ability to make comparative judgments about the attributes or parameter values of facts [25]. For models trained on data generated from functions considered ID, we evaluate whether these models can generalize to the same data-generating function with larger or smaller parameter values that are considered OOD. Specifically, we define the datasets as follows: $\mathcal{P}_{\text{train}}, \mathcal{P}_{\text{test}} \subset \mathcal{F}_{sc}$, where $\mathcal{P}_{\text{train}} \cap \mathcal{P}_{\text{test}} = \emptyset$. Consider a specific example:

$$f_1 \in \mathcal{P}_{\text{train}}, f_2 \in \mathcal{P}_{\text{test}}, \forall x \in \mathbb{R}, (x, f_1, y) \in \mathcal{P}_{\text{train}} \text{ and } (x, f_2, y) \in \mathcal{P}_{\text{test}}. \quad (2)$$



(a) Composition (b) Comparison (c) Inverse Search

Figure 1: Implicit reasoning tasks for time series include composition (a), comparison (b), and inverse search (c). In composition, models predict new composite series seen only at inference. In comparison, they forecast series from unseen function values. In inverse search, models predict decomposed component series. The black dashed line in the figure indicates in-distribution time series observed during training (left) and out-of-distribution data observed during inference (right).

Table 1: Synthetic dataset functions and parameters for implicit reasoning tasks.

Task	Dataset	Function	Parameters
Composition	$\mathcal{P}_{\text{train}}$	$f_s \in \mathcal{F}_s, f_c \in \mathcal{F}_c$	
Addition	$\mathcal{P}_{\text{test}}$	$f_s + f_c \in \mathcal{F}_{sc}$	$a=0 \wedge b \in [b_l, b_u] \wedge c=0 \wedge m \in [m_l, m_u]$
Subtraction	$\mathcal{P}_{\text{test}}$	$f_s - f_c \in \mathcal{F}_{sc}$	
Multiplication	$\mathcal{P}_{\text{test}}$	$f_s \times f_c \in \mathcal{F}_{sc}$	
Composition	$\mathcal{P}_{\text{train}}$	$f_s + f_c \in \mathcal{F}_{sc}$	$a \in [a_l, a_u] \vee b \in [b_l, b_u] \vee c \in [c_l, c_u] \vee m \in [m_l, m_u]$
Function	$\mathcal{P}_{\text{test}}$	$f_s + f_c \in \mathcal{F}_{sc}$	$a \in [a_l, a_u] \wedge b \in [b_l, b_u] \wedge c \in [c_l, c_u] \wedge m \in [m_l, m_u]$
Comparison	$\mathcal{P}_{\text{train}}$	$f_s + f_c \in \mathcal{F}_{sc}$	$a \in [a_l, a_u] \vee b \in [b_l, b_u] \vee c \in [c_l, c_u] \vee m \in [m_l, m_u]$
	$\mathcal{P}_{\text{test}}$	$f_s + f_c \in \mathcal{F}_{sc}$	$i < i_l \vee i > i_u$ for $i=a \vee i=b \vee i=c \vee i=m$
Inverse Search	$\mathcal{P}_{\text{train}}$	$f_s + f_c \in \mathcal{F}_{sc}$	$a=0 \wedge b \in [b_l, b_u] \wedge c=0 \wedge m \in [m_l, m_u]$
	$\mathcal{P}_{\text{test}}$	$f_s \in \mathcal{F}_s, f_c \in \mathcal{F}_c$	

We train models on signals with one varying parameter: $a, b, c,$ or m ($n = 1200$) and evaluate them on signals with smaller or larger parameter values not seen during training ($n = 120$).

Inverse search. We evaluate whether models can decompose functions through the inverse search task. For models trained on signal compositions, we assess their ability to accurately forecast individual basis functions comprising the signal, which are considered OOD. We define the datasets as follows: $\mathcal{P}_{\text{train}} = \mathcal{F}_{sc}$ and $\mathcal{P}_{\text{test}} = \mathcal{F}_s \cup \mathcal{F}_c$, where $\mathcal{P}_{\text{train}} \cap \mathcal{P}_{\text{test}} = \emptyset$. Consider a specific example:

$$f_s \in \mathcal{F}_s, f_c \in \mathcal{F}_c, f_{sc} \in \mathcal{F}_{sc}, \forall x \in \mathbb{R}, (x, f_{sc}, y) \in \mathcal{P}_{\text{train}} \text{ and } (x, f_s, y), (x, f_c, y) \in \mathcal{P}_{\text{test}} \quad (3)$$

We train models on composite addition of trend and seasonality series ($n = 900$). We then evaluate the models on all decomposed trend and seasonality components series ($n = 60$).

3.2 Data

We use the functions in Table 1, with parameter values detailed in Appendix A.3 to generate synthetic data. Both ID and OOD data use $x \in [0, 1]$ with 1200 samples. Models are trained on the first 1000 ID samples and evaluated on the last 200 OOD samples, testing their implicit reasoning capabilities.

3.3 Models

We trained models using 13 algorithm implementations obtained from the `Neuralforecast` [19] library. Trained models include: Multi-Layer Perceptron (`MLP`) [21], `DLinear` [31], `NHITS` [5], `TSMixer` [6], Long-Short Term Memory (`LSTM`) [22], Temporal Convolution Network (`TCN`) [4, 23], `TimesNet` [29], `VanillaTransformer` [24, 33], Temporal Fusion Transformer (`TFT`) [14], `Autoformer` [28], `Informer` [33], inverted Transformer (`iTransformer`) [15], and patch time series

Table 2: Mean Absolute Error (MAE) computed across all forecast points within the specified time horizon and averaged across series for each task. Best results are in bold, second-best in blue.

Model	Composition				Comparison	Inverse Search
	Add.	Sub.	Mult.	Func.		
DLinear	**0.689**	**0.935**	10.522	21.155	**1.072**	0.539
NHITS	1.427	2.36	**2.065**	**3.868**	2.218	**0.214**
VanillaTransformer	2.432	2.993	8.249	11.291	2.120	0.551
TFT	1.024	3.243	13.395	10.159	2.045	0.574
Autoformer	1.708	3.999	12.845	13.383	3.966	1.180
Informer	2.045	3.025	9.246	12.268	2.131	0.877
iTransformer	1.358	2.994	15.197	31.632	2.885	1.164
PatchTST	1.717	2.168	3.497	6.209	1.740	0.394

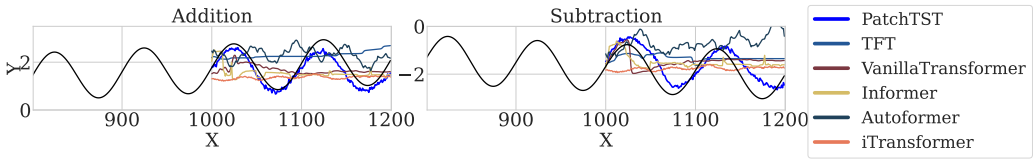


Figure 2: PatchTST shows capability in generating forecasts that accurately capture the composition of trend and seasonality series compared to other Transformer models.

Transformer (PatchTST) [18]. This controlled setup is crucial for directly comparing architectures under identical conditions. By evaluating Transformer models with key components like attention and time-series patching, aligned to various TSFMs, our work delivers valuable insights into how TSFMs may leverage implicit reasoning in OOD generalization. We also generate zero-shot forecasting results for pretrained Chronos, LagLlama, TimesFM, and MOMENT. More information on these models, as well as model training and hyperparameters, is provided in Appendix A.2 and A.4, respectively.

4 Results

PatchTST outperformed all other evaluated Transformer models on the composition task, particularly for addition and subtraction tasks as shown in Fig 2. However, the model's performance on these tasks is sensitive to patch length, as shown in Fig. 3a in Appendix B, indicating the critical role of patch length in the model's knowledge manipulation capabilities. Table 2 includes Mean Absolute Error (MAE) computed across all points within the forecast horizon and averaged across series for each task for DLinear, NHITS, and Transformer models. As shown in Table 2, DLinear had the smallest forecasting error for addition and subtraction composition tasks as well as comparison tasks, while NHITS had the smallest error for multiplication composition, function composition, and inverse search tasks. Results for each task, including standard deviations, for all 15 models are shown in Tables 4, 5a, and 5b in Appendix B. These tables also include baseline results for models trained on the first 1000 samples of OOD data. Example forecasts for composition and inverse search tasks are shown in Figs. 3 and 4 in Appendix B. MAE formulation is provided in A.5.

5 Discussion

Our findings suggest that linear, MLP-based, and patch-based Transformer models, like PatchTST, offer implicit reasoning capabilities beyond memorization. PatchTST demonstrates strong OOD forecasting performance, supporting the use of patching input time series in TSFMs like MOMENT, Moirai, and TimesFM. Moreover, NHITS's robust performance across tasks highlights the value of signal decomposition and hierarchical architectures, which if incorporated in future TSFMs could potentially enhance their forecasting capabilities. Our evaluation also shows the need for metrics beyond traditional loss measures. For example, despite TFT's second-best MAE in addition composition, it forecasts a straight line as shown in Fig. 2, demonstrating the limitations of loss functions in capturing forecast pattern quality in complex tasks.

Acknowledgment

This work was partially supported by the NSF (awards 2406231 and 2427948), NIH (awards R01NS124642 and R01DK131586), DARPA (HR00112420329), and the US Army (W911NF-20-D0002). We would also like to thank Kin Gutierrez Olivares for his feedback on our manuscript.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. 2023.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. In *Proceedings of the 41st International Conference on Machine Learning*, page 235, 2024.

[3] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. 2024.

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[5] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-HiTS: Neural hierarchical interpolation for time series forecasting. In *AAAI-23*, 2022.

[6] Si-An Chen, Chun-Liang Li, Nathanael C. Yoder, Sercan Ö. Arık, and Tomas Pfister. TSMixer: An all-MLP architecture for time series forecasting. In *Published in Transactions on Machine Learning Research*, 2023.

[7] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. 2024.

[8] Vijay Ekambaram, Arindam Jati, Nam H Nguyen, Pankaj Dayama, Chandra Reddy, Wesley M Gifford, and Jayant Kalagnanam. TTMs: Fast Multi-level Tiny Time Mixers for Improved Zero-shot and Few-shot Forecasting of Multivariate Time Series. *arXiv preprint arXiv:2401.03955*, 2024.

[9] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biol. Cybernetics*, 20:121—136, 1975.

[10] Azul Garza and Max Mergenthaler-Canseco. TimeGPT-1, 2023.

[11] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *41st International Conference on Machine Learning*, 2024.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, 2017.

[13] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[14] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4): 1748–1764, 2021.

[15] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting, 2024.

[16] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative Pre-trained Transformers Are Large Time Series Models. In *Forty-first International Conference on Machine Learning*, 2024.

[17] Vinod Nair and Jeoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML-23*, 2010.

[18] Yuqi Nie, Nam H. Nguyen, and Phanwadee Sinthong an Jayant Kalagnanam2. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[19] Kin G. Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022, 2022. URL https://github.com/Nixtla/neuralforecast.

[20] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards foundation models for probabilistic time series forecasting, 2024.

[21] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386—-408, 1958.

[22] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

[23] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio, 2016.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, and Aidan N. Gomez. Attention is all you need, 2017.

[25] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. 2024.

[26] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[27] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[28] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2021.

[29] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *Proceedings of the 34th International Conference on Learning Representations*, 2023.

[30] Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

[31] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[32] Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*, 2023.

[33] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.

A Supplemental Information

A.1 Extended Related Work

Implicit Reasoning in LLMs. Prior research has explored implicit reasoning in language models [1, 2, 25, 30, 32]. Implicit reasoning is often assessed through tasks that require models to apply knowledge learned during training to new test instances. One common form is *composition*, where models must chain multiple facts to answer a question [25, 30, 32]. Other forms of implicit reasoning explored include *comparison* and *inverse search* [1, 25]. Comparison involves models evaluating two or more entities to make judgments, such as determining whether the attribute value of one entity is greater or smaller than that of another. Inverse search tests a model's ability to generate predictions in the reverse order of the training task. For example, this could involve applying the model to identify an entity based on its attributes when it was originally trained to predict the attributes of entities. Allen-Zhu et al. found that generative models struggle with inverse search unless pretrained for it specifically [1].

While models can recall individual facts well, they struggle with multi-hop reasoning that requires 'chain-of-thought' logic. Yang et al. [30] found evidence of latent multi-hop reasoning for specific fact compositions, but noted it is highly contextual. Wang et al. demonstrated that Transformers can learn implicit reasoning, but only after extended training beyond typical overfitting thresholds [25]. Composition tasks have also been studied in machine translation, testing whether models can generalize learned command components to new conjunctions, such as repeating actions [13].

A.2 Models

Multi Layer Perceptrons (`MLP`) - A neural network architecture composed of stacked Fully Connected Neural Networks trained with backpropagation [17, 9, 21].

Neural Hierarchical Interpolation for Time Series (`NHITS`) - A deep learning model that applies multi-rate input pooling, hierarchical interpolation, and backcast residual connections together to generate additive predictions with different signal bands [5].

Long Short-Term Memory Recurrent Neural Network (`LSTM`) - A recurrent neural network (`RNN`) architecture that transforms hidden states from a multi-layer `LSTM` encoder into contexts which are used to generate forecasts using `MLP`s [22].

Temporal Convolution Network (`TCN`) - A 1D causal-convolutional network architecture that transforms hidden states into contexts which are used as inputs to `MLP` decoders to generate forecasts. Causal convolutions are used to generate contexts by convolving the prediction at time t only with elements from time t and earlier [4, 23].

TimesNet (`TimesNet`) - A deep learning architecture that transforms the original 1D time series into a set of 2D tensors based on multiple periods to capture intra- and inter-period variations modeled by 2D kernels [29].

VanillaTransformer (`VanillaTransformer`) - An encoder-decoder architecture with a multi-head attention mechanism that uses autoregressive features from a convolution network, window-relative positional embeddings from harmonic functions, and absolute positional embeddings from calendar data. An MLP decoder outputs time series predictions in a single pass [24, 33].

Temporal Fusion Transformer (`TFT`) - An attention-based deep learning architecture that learns temporal relationships at different scales using `LSTM`s for local processing and self-attention layers to model long-term dependencies. It also leverages variable selection networks and a series of gating layers to suppress unnecessary processing in the architecture [14].

Autoformer (`Autoformer`) - An encoder-decoder architecture with a multi-head attention mechanism that uses autoregressive features from a convolution network and absolute positional embeddings from calendar data. Decomposed trend and seasonal components are obtained using a moving average filter and an Auto-Correlation mechanism is used to identify periodic dependencies and aggregate similar sub-series [28, 24].

Informer (`Informer`) - An encoder-decoder architecture with a multi-head attention mechanism that has three key features: a ProbSparse self-attention mechanism with $O(L \log L)$ complexity, a self-attention distilling process, and an MLP decoder that outputs time-series predictions in a single pass.

It uses autoregressive features from a convolution network, window-relative positional embeddings from harmonic functions, and absolute positional embeddings from calendar data [33, 24].

iTransformer (`iTransformer`) - An attention-based deep learning architecture that applies attention and feed-forward networks to inverted dimensions by embedding time points into variate tokens. The attention mechanism capture multivariate correlations while the feed-forward network learns nonlinear representations for each token [15].

Patch Time Series Transformer (`PatchTST`) - An encoder-only, multi-head attention-based architecture that separates input time series into sub-series level patches as input tokens. Each channel is dedicated to a single univariate time series, and all channels use the same embedding and Transformer weights. [18]

LagLlama (`LagLlama`) [20] - A foundation model for univariate probabilistic time series forecasting based on a decoder-only Transformer architecture that uses lags as covariates [20]. We use pretrained model weights from the HuggingFace library [26]. We use a context length of 32, as this is the parameter value on which the model was trained. Although the model can handle other context lengths, which may improve forecasting performance.

Chronos (`Chronos`) - A family of pretrained time series forecasting models that creates token sequences through scaling and quantization for input to a language model. Probabilistic forecasts are generated by sampling future trajectories from historical data [3]. We use the 't5-efficient-small' pretrained model weights from the HuggingFace library [26].

MOMENT (`MOMENT`) - `MOMENT`is a family of foundation models for time series that uses an attention-based architecture and processes input time series into fixed-length patches with each mapped to a D-dimensional embedding. During pretraining, patches are masked to minimize reconstruction error, and a separate linear projection is trained to adapt embeddings for long-horizon forecasting and other tasks [11]. We use the 'MOMENT-1-large' pretrained model weights from the HuggingFace library [26].

TimesFM (`TimesFM`) - An attention-based decoder-only Transformer architecture that leverages input time series patching, masked causal attention, and sequential patch output predictions, with flexible output patch size [7]. We use the 'timesfm-1.0-200m' pretrained model weights from the HuggingFace library [26].

A.3 Synthetic Data Parameters

We leverage open-source code from `MOMENT` [11] to generate synthetic sinusoidal time series with varying trend, frequency, baseline, and amplitude. We use the following parameter values for all implicit reasoning tasks: $a \in [1, 32]$, $b \in [3, 32]$, $c \in [-32, 32]$. $m \in [1, 32]$ was used for addition, subtraction, and multiplication composition tasks, as well as for inverse search tasks. $m \in [-32, 32]$ was used for function composition and comparison tasks. For composition tasks, we generate $n = 30$ in-distribution (ID) series with evenly spaced parameter values. For comparison tasks, we generate $n = 300$ ID series with evenly spaced parameter values. For the inverse search task, we generate $n = 900$ ID addition composition aggregates using $n = 30$ trend and seasonality series, each with evenly spaced parameter values.

A.4 Model Training and Parameters

Models were trained using `Adam` optimizer [12]. We train models with the following parameters outlined in Table 3 to ensure consistent evaluation across architectures. The training loss function used was Mean Squared Error (MSE), where H refers to the forecast horizon, t refers to the time point at which forecasts are generated, y refers to the target signal values, and $\hat{y}$ refers to the model's predicted values:

$$MSE(y_{t+1,t+H}, \hat{y}_{t+1,t+H}) = \frac{1}{H} \sum_{i=t+1}^{t+H} (y_i - \hat{y}_i)^2.$$

The deep learning models were trained using an NVIDIA A100 Tensor Core GPU.

Table 3: Common hyperparameter search space

Hyperparameter	Considered Values
Input size	200
Learning rate	1e-3
Batch size	4
Windows batch size	256
Dropout	0.0
Training steps	2000
Validation check steps	50
Early stop patience steps	5
Random seed	0
Patch Length	DiscreteRange(1, 50, 100, 150, 200)
Hidden size	128
Model layers	3

A.5 Evaluation Metrics

We use **Mean Absolute Error (MAE)** to evaluate model performance. Here, H refers to the forecast horizon, t refers to the time point at which forecasts are generated, y refers to the target signal values, and $\hat{y}$ refers to the model's predicted values:

$$MAE(y_{t+1,t+H}, \hat{y}_{t+1,t+H}) = \frac{1}{H} \sum_{i=t+1}^{t+H} |y_i - \hat{y}_i|.$$

A.6 Broader Impacts

This work aims to enhance our understanding of whether time series models generalize well by learning the underlying concepts of temporal dynamics or merely memorizing specific patterns seen during training. We find that certain models generalize effectively in systematically orchestrated out-of-distribution scenarios, suggesting underexplored reasoning capabilities beyond simple pattern memorization. Since this research is foundational and does not involve direct applications, we do not anticipate any negative societal impacts. Instead, our findings may provide valuable insights for developing more data- and computationally efficient deep learning architectures. Additionally, our results may help define the limitations of time series models, ensuring they are not applied in contexts where poor generalization performance is anticipated.

A.7 Limitations

This study has several limitations that should be acknowledged. First, our analysis relies on synthetic data, which may not fully capture the complexities and nuances present in real-world datasets. While synthetic data allows for controlled experimentation, it may limit the generalizability of our findings to broader applications. Second, we employed consistent hyperparameters across models to ensure a fair evaluation of performance; however, different combinations of hyperparameters may yield improved results. Future work will explore the impact of hyperparameter variations on model performance.

A.8 Reproducibility Statement

All models are open-source. Models trained on synthetic data can be found in the `Neuralforecast` library [19]. Pretrained foundation models including, `LagLlama` [20], `Chronos` [3], `TimesFM` [7], and `MOMENT` [11], are open-source, with model weights available in the Hugging Face library [26]. All models were trained and evaluated on a computing cluster consisting of 128 AMD EPYC 7502 CPUs, 503 GB of RAM, and 8 NVIDIA RTX A6000 GPUs each with 49 GiB RAM. The synthetic datasets used in our study will be released publicly with the full paper.

B Supplemental Results

Table 4: Mean Absolute Error (MAE) averaged across series (standard deviation) for the *composition* implicit reasoning tasks: addition (add.), subtraction (sub.), multiplication (mult.), and sinusoidal function (function). MAE results for models trained on in-distribution (ID) time series and evaluated on out-of-distribution (OOD) series are presented in the 'Task' column. We also include baseline (base.) MAE results for models that are both trained and evaluated on OOD data. The best results are highlighted in bold, and the second best results are highlighted in blue.

Model	**MAE**							
	Add.		**Sub.**		**Mult.**		**Function**	
	Task	**Base.**	**Task**	**Base.**	**Task**	**Base.**	**Task**	**Base.**
MLP	1.604 (1.173)	0.386 (0.284)	2.637 (1.648)	0.350 (0.197)	2.097 (1.559)	1.507 (1.127)	10.099 (6.372)	2.730 (2.3)
DLinear	**0.689** (0.349)	0.743 (0.333)	**0.935** (0.452)	0.715 (0.333)	10.522 (7.945)	8.792 (5.284)	21.155 (23.965)	10.653 (6.783)
NHITS	1.427 (1.089)	**0.216** (0.132)	2.36 (1.489)	**0.184** (0.215)	**2.065** (1.687)	**1.194** (0.813)	**3.868** (2.207)	**1.736** (1.547)
TSMixer	1.398 (0.766)	1.185 (0.511)	4.862 (2.167)	1.348 (0.651)	16.609 (9.531)	11.427 (7.569)	20.943 (16.653)	13.008 (7.985)
LSTM	7.546 (4.288)	2.973 (1.793)	8.992 (4.954)	5.216 (2.904)	9.831 (5.529)	5.641 (3.714)	11.353 (5.12)	5.095 (2.928)
TCN	5.716 (3.228)	2.538 (1.432)	5.063 (2.597)	3.057 (1.667)	8.359 (4.816)	2.606 (1.995)	11.413 (4.781)	4.123 (2.391)
TimesNet	1.630 (0.627)	0.555 (0.238)	2.937 (1.519)	0.512 (0.239)	10.635 (7.88)	2.082 (1.692)	11.643 (8.352)	3.744 (3.076)
VanillaTransformer	2.432 (1.468)	0.721 (0.228)	2.993 (1.69)	0.658 (0.152)	8.249 (4.939)	5.032 (3.492)	11.291 (7.334)	7.341 (4.552)
TFT	1.024 (0.408)	0.709 (0.295)	3.243 (1.507)	1.057 (0.504)	13.395 (9.688)	5.121 (3.834)	10.159 (5.228)	6.548 (4.455)
Autoformer	1.708 (0.952)	0.914 (0.315)	3.999 (1.886)	1.188 (0.52)	12.845 (7.764)	9.69 (5.716)	13.383 (8.018)	11.344 (6.846)
Informer	2.045 (1.273)	0.741 (0.235)	3.025 (1.702)	0.78 (0.28)	9.246 (5.303)	5.423 (3.4)	12.268 (8.89)	10.335 (5.724)
iTransformer	1.358 (0.886)	0.933 (0.384)	2.994 (1.403)	0.947 (0.447)	15.197 (10.35)	14.618 (8.732)	31.632 (16.823)	11.773 (6.882)
PatchTST	1.717 (1.113)	0.435 (0.285)	2.168 (1.185)	0.584 (0.404)	3.497 (2.442)	1.600 (1.108)	6.209 (4.009)	7.231 (6.031)
TSFMs	**Zero-shot Forecasting**							
LagLlama	5.181 (2.720)		4.263 (2.475)		9.904 (5.795)		17.684 (7.206)	
Chronos	1.064 (1.133)		1.124 (1.228)		1.922 (2.499)		2.332 (2.596)	
TimesFM	0.475 (0.375)		0.467 (0.457)		0.355 (0.336)		0.672 (0.655)	
MOMENT	2.428 (1.392)		3.140 (1.564)		9.960 (6.022)		10.877 (6.347)	

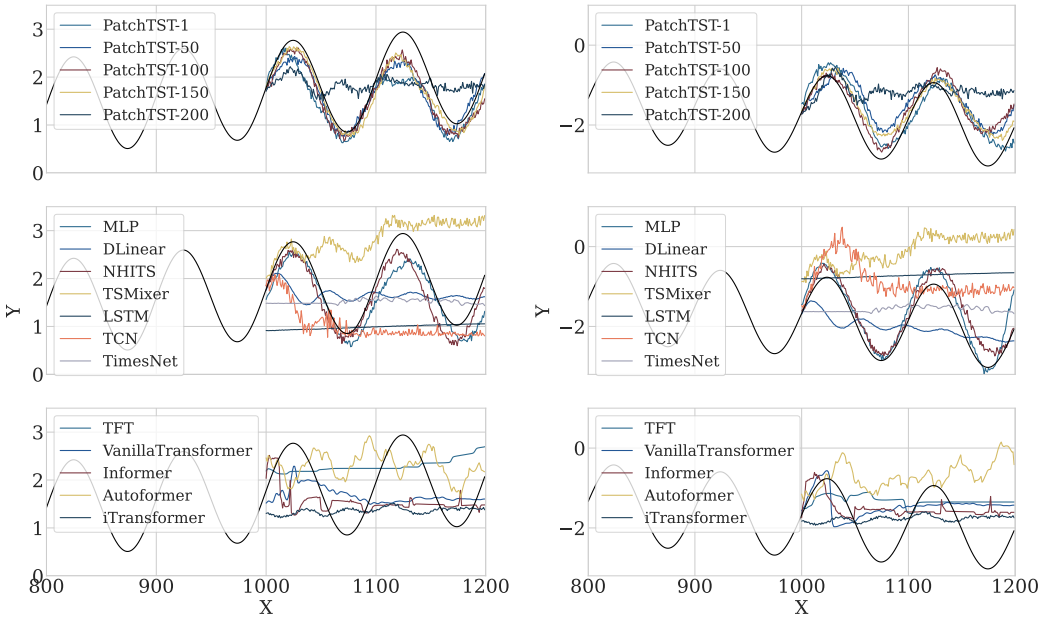
Table 5: Model forecast Mean Absolute Error (MAE) averaged across series (standard deviation). MAE results for models trained on in-distribution (ID) time series and evaluated on out-of-distribution (OOD) series are presented in the 'Task' column. We also include baseline (base.) MAE results for models that are both trained and evaluated on OOD data. The best results are highlighted in bold, and the second best results are highlighted in blue.

(a) *Comparison* Task

Model	**MAE**	
	Comparison	
	Task	**Base.**
MLP	2.492 (4.831)	1.945 (4.138)
DLinear	**1.072** (1.173)	1.342 (1.848)
NHITS	2.218 (4.113)	1.750 (3.749)
TSMixer	2.488 (3.371)	2.391 (3.570)
LSTM	3.174 (3.767)	4.663 (5.571)
TCN	2.467 (3.002)	2.913 (3.591)
TimesNet	3.570 (6.309)	1.972 (4.001)
VanillaTransformer	2.120 (4.071)	2.040 (4.066)
TFT	2.045 (3.802)	1.986 (3.864)
Autoformer	3.966 (5.097)	3.785 (5.145)
Informer	2.131 (3.805)	1.999 (3.674)
iTransformer	2.885 (4.052)	3.106 (4.429)
PatchTST	1.740 (3.084)	**1.204** (2.678)
TSFMs	**Zero-Shot**	
LagLlama	5.307 (7.512)	
Chronos	0.704 (1.362)	
TimesFM	0.413 (0.775)	
MOMENT	3.381 (4.557)	

(b) *Inverse Search* Task

Model	**MAE**	
	Inverse Search	
	Task	**Base.**
MLP	0.632 (0.523)	0.076 (0.040)
DLinear	0.539 (0.433)	0.444 (0.389)
NHITS	**0.214** (0.231)	**0.035** (0.055)
TSMixer	1.622 (0.697)	1.431 (0.647)
LSTM	1.784 (1.617)	4.161 (4.665)
TCN	1.666 (1.444)	3.113 (3.504)
TimesNet	0.646 (0.384)	0.438 (0.385)
VanillaTransformer	0.551 (0.531)	0.279 (0.248)
TFT	0.574 (0.464)	0.961 (0.504)
Autoformer	1.180 (0.527)	0.804 (0.335)
Informer	0.877 (0.425)	0.436 (0.210)
iTransformer	1.164 (0.507)	1.143 (0.560)
PatchTST	0.394 (0.545)	0.293 (0.195)
TSFMs	**Zero-Shot**	
LagLlama	2.656 (2.724)	
Chronos	0.096 (0.088)	
TimesFM	0.070 (0.075)	
MOMENT	1.502 (1.225)	



(a) Addition (b) Subtraction

Figure 3: Model time series forecasts for addition (a) and subtraction (b) *composition* tasks.

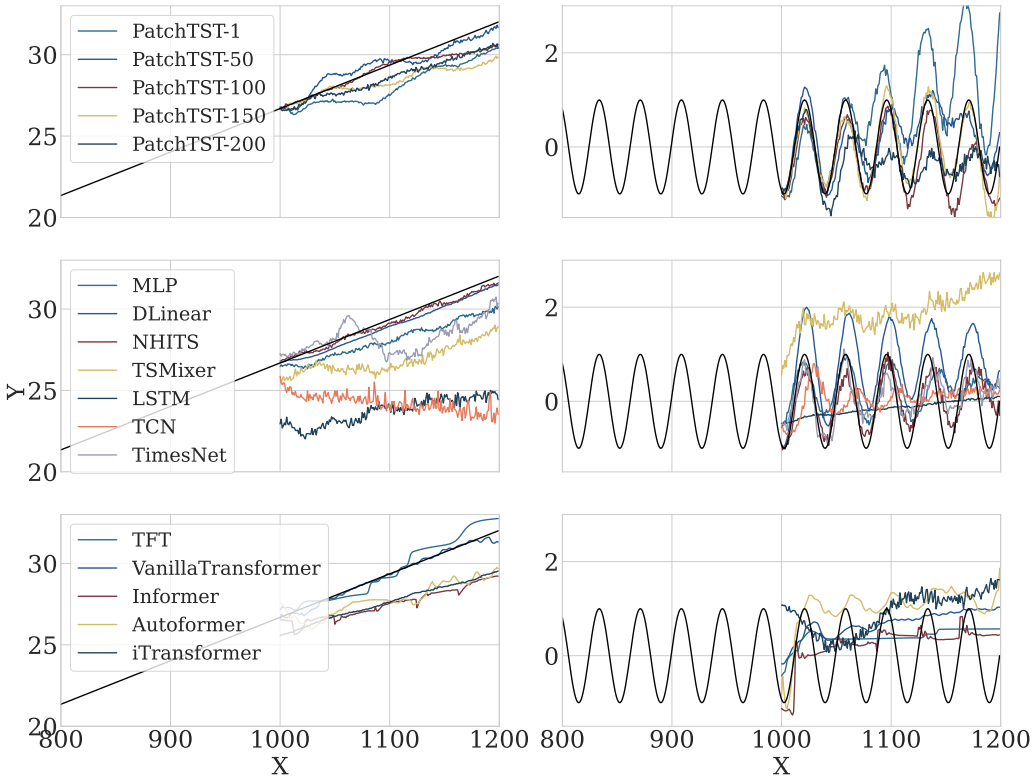


Figure 4: Model time series forecasts for the *inverse search* task.