

## A Related Work

**Time Series Forecasting Models** Time series forecasting has evolved through several paradigms, from classical statistical methods to advanced deep learning approaches. Traditional statistical methods like ARIMA [1], ETS [2], and VAR [3] provided interpretable frameworks based on explicit statistical assumptions but struggled with complex non-linear patterns. Machine learning approaches, including XGBoost [4, 5], Random Forests [6], and LightGBM [7], offered improved flexibility but often required manual feature engineering. The deep learning revolution brought a diverse array of architectures for time series modeling, progressing from MLP-based models (N-BEATS [8], DLinear [9], N-HiTS [10], NLinear [9]) to RNN architectures [11, 12, 13, 14] (LSTM [15, 16], GRU [17, 18, 19], DeepAR [20], SegRNN [21]) and CNN architectures (TCN [22], WaveNet [23], SCINet [24], TimesNet [25]), and then to transformer-based models (Informer [26], Autoformer [27], PatchTST [28], FEDformer [29], Pyraformer [30], iTransformer [31]). Recent advances include specialized architectures like TimesNet [25] for periodic pattern modeling, TSMixer [32] and TimeMixer [33] for efficient temporal mixing, and filter-based approaches (PaiFilter [34], TexFilter [34]). Meanwhile, foundation models for time series have emerged, including Chronos [35], Lag-Llama [36], TimesFM [37], Timer [38], MOIRAI [39], UniTS [40], and TimeMoE [41], offering promising zero-shot forecasting capabilities across diverse domains.

**Time Series Benchmarks and Evaluation** A variety of benchmarks have been proposed for time series forecasting, such as M3 [42], M4 [43], Monash [44], LTSF-Linear [9], BasicTS [45], and BasicTS+ [46]. However, these benchmarks are not comprehensive: some focus only on univariate or multivariate forecasting, and many either omit statistical methods or deep learning methods, resulting in limited coverage of different modeling paradigms. Recently, more comprehensive benchmarks have emerged to carry out more detailed evaluations. ProbTS [47] advances probabilistic forecasting evaluation with improved uncertainty quantification, TSlib [48] covers multiple time series tasks including forecasting, anomaly detection, and missing value imputation within a unified pipeline, and TFB [49] provides fine-grained dataset categorization with scalable integration of diverse methods. Despite these improvements, existing benchmarks still struggle to systematically isolate specific temporal patterns, establish theoretical performance boundaries, or provide interpretable capability mapping. Our SynTSBench framework is designed to fill these gaps by enabling controllable synthetic data generation, systematic decomposition of temporal features, and rigorous, interpretable evaluation of model capabilities across a wide range of forecasting scenarios. As shown in Table 1, this table presents a comparison between SynTSBench and other benchmarks.

Table 1: Time series forecasting benchmark comparison. ✓ indicates present, ✗ indicates absent, — indicates incomplete.

Benchmark	Univariate Forecasting	Multivariate Forecasting	Patterns-learning Assessment	Theoretical Performance Boundaries	Cross-variable Relationship Assessment	Robustness Testing	Foundation Models Zero-shot Testing
M3 [42]	✓	✗	✗	✗	✗	✗	✗
M4 [43]	✓	✗	✗	✗	✗	✗	✗
TSlib [48]	✓	✓	✗	✗	✗	—	✗
BasicTS [45]	✗	✓	✗	✗	✗	✗	✗
BasicTS+ [46]	✗	✓	—	✗	✗	✗	✗
Monash [44]	✓	✗	—	✗	✗	✗	✗
ProbTS [47]	✓	✓	—	✗	✗	✗	✓
TFB [49]	✓	✓	—	✗	✗	✗	✗
SynTSBench (Ours)	✓	✓	✓	✓	✓	✓	✓

## B Additional Experiments and Supplementary Results

We provide additional experiments and results to further validate SynTSBench. Appendix B.1 evaluates the effect of dataset length on model performance. Appendix B.2 presents results on complex real-world pattern simulations. Appendix B.3 examines zero-shot forecasting capabilities of foundation models. Appendix B.4 investigates the impact of normalization layers on model performance.

### B.1 Length of dataset

In this experiment, we evaluated the performance of various models on datasets of different lengths: 1000, 5000, 10000, and 20000. Figure 1 shows the average MSE for each model as the dataset length increases. It is evident that most models benefit from longer datasets, with their MSE values decreasing as more data becomes available.

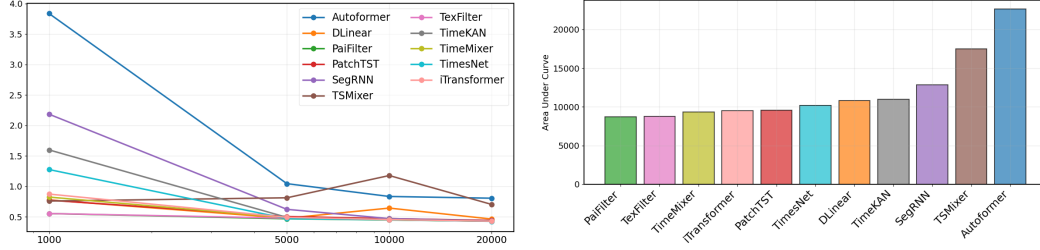


Figure 1: Average MSE by Dataset Length for Figure 2: Model Efficiency Ranking (Lower AUC means better performance)

To further quantify the efficiency of each model in learning from datasets of varying lengths, we calculated the area under the curve (AUC) formed by each model’s MSE curve and the x-axis. A smaller AUC indicates better efficiency in learning from the data. Figure 2 ranks the models based on their AUC values, where lower values correspond to higher efficiency.

The results demonstrate that models based on FilterNet, specifically PaiFilter and TexFilter, achieved the best performance, as they consistently exhibited lower MSE values and smaller AUCs compared to other models. This highlights their superior ability to learn effectively from datasets of varying lengths.

## B.2 Performance on Complex Real-world Pattern Simulations

Table 2: MSE and MAE Results for Univariate Datasets. **Red bold** indicates best performance, **blue underlined** indicates second-best performance. The following tables use the same notation.

Dataset	Metric	Model											Optimal	
		Autoformer	DLinear	PaiFilter	PatchTST	SegRNN	TSMixer	TexFilter	TimeKAN	TimeLLM	TimeMixer	TimesNet		iTransformer
Economic-Indicator	MSE	0.1017	0.1017	0.0721	0.0603	0.0617	0.8370	0.0548	<b>0.0534</b>	0.0547	0.0545	<b>0.0534</b>	0.0607	<b>0.0424</b>
	MAE	0.2564	0.2453	0.2095	0.1989	0.1984	0.7999	0.1885	<b>0.1836</b>	0.1876	0.1878	<b>0.1872</b>	0.1992	<b>0.1609</b>
Electricity-Consumption	MSE	0.3593	0.1921	0.1201	0.1269	0.1719	0.1298	<b>0.1183</b>	0.1523	0.1605	0.1324	0.1243	<b>0.1189</b>	<b>0.0833</b>
	MAE	0.4494	0.3358	<b>0.2426</b>	0.2571	0.3203	0.2630	0.2427	0.2897	0.3036	0.2592	0.2450	<b>0.2414</b>	<b>0.1851</b>
Industrial-Sensor	MSE	0.1027	0.0227	0.0188	0.0289	0.2849	0.0270	0.0187	0.0191	0.0350	0.0189	<b>0.0172</b>	<b>0.0172</b>	<b>0.0151</b>
	MAE	0.2533	0.1199	0.1085	0.1353	0.4099	0.1307	0.1083	0.1098	0.1487	0.1090	<b>0.1049</b>	<b>0.1037</b>	<b>0.0974</b>
Industrial-Sensor-Normal	MSE	0.2441	0.0608	<b>0.0533</b>	0.0590	0.1643	0.0594	<b>0.0542</b>	0.0556	0.0694	0.0547	0.0552	0.0544	<b>0.0495</b>
	MAE	0.3165	0.1963	<b>0.1831</b>	0.1921	0.3257	0.1931	<b>0.1847</b>	0.1870	0.2092	0.1854	0.1864	<b>0.1847</b>	<b>0.1772</b>
Network-Traffic	MSE	1.3672	1.1490	0.8882	0.9019	1.2382	1.1145	<b>0.8804</b>	0.8855	<b>0.8644</b>	0.9037	0.8904	0.9277	<b>0.8129</b>
	MAE	0.7985	0.6810	0.5594	0.5416	0.7490	0.6427	<b>0.5493</b>	<b>0.5083</b>	<b>0.5288</b>	0.5368	0.5555	0.5326	<b>0.5413</b>
Retail-Sales	MSE	0.7671	<b>0.4314</b>	0.4715	0.4549	0.6358	0.4423	0.4765	<b>0.4005</b>	0.5329	0.4379	0.4919	0.5001	<b>0.2708</b>
	MAE	0.7087	0.4539	0.5083	0.4861	0.6312	<b>0.4434</b>	0.5096	<b>0.4454</b>	0.5599	0.4562	0.5145	0.5147	<b>0.2535</b>
Stock-Price	MSE	0.1933	0.1898	0.1812	0.1794	0.1774	0.1815	0.1779	<b>0.1660</b>	0.1742	<b>0.1663</b>	0.1825	0.1797	<b>0.1199</b>
	MAE	0.3460	0.3424	0.3336	0.3327	0.3308	0.3330	0.3301	<b>0.3213</b>	0.3271	<b>0.3182</b>	0.3347	0.3317	<b>0.2708</b>
Temperature-Sensor	MSE	0.1750	0.1547	<b>0.0628</b>	0.0703	0.0772	0.1549	0.0653	0.0673	0.1101	0.0653	0.0641	<b>0.0639</b>	<b>0.0508</b>
	MAE	0.3357	0.3186	<b>0.1974</b>	0.2110	0.2211	0.3245	0.2018	0.2056	0.2718	0.2020	0.2001	<b>0.1996</b>	<b>0.1768</b>
Website-Traffic	MSE	0.4310	0.0799	0.0531	0.0592	0.1344	0.0677	<b>0.0509</b>	0.0782	0.0913	0.0682	0.0595	<b>0.0445</b>	<b>0.0296</b>
	MAE	0.5519	0.2255	0.1846	0.1959	0.2931	0.2073	<b>0.1805</b>	0.2252	0.2451	0.2088	0.1951	<b>0.1686</b>	<b>0.1402</b>

Table 3: MSE and MAE results for multivariate datasets

Dataset	Metric	Model											
		Autoformer	DLinear	PaiFilter	PatchTST	SegRNN	TSMixer	TexFilter	TimeKAN	TimeLLM	TimeMixer	TimesNet	iTransformer
Ad-Sales	MSE	1.4331	1.2836	1.2681	1.2638	1.3543	1.2483	1.2834	1.3236	1.4684	1.2509	1.3167	1.3274
	MAE	0.8169	0.7306	0.7194	0.7191	0.7681	0.7069	0.7278	0.7486	0.8264	0.7050	0.7400	0.7395
Intervention-Effect	MSE	0.5306	0.1174	0.1026	0.1072	0.0994	0.1208	0.1064	0.1008	0.1126	0.1012	0.0980	0.1010
	MAE	0.4202	0.2509	0.2238	0.2286	0.2214	0.2553	0.2287	0.2242	0.2368	0.2227	0.2198	0.2254
Lotka-Volterra	MSE	1.8547	0.1660	0.0929	0.1073	0.0233	0.0068	0.0198	0.0978	0.2453	0.0158	0.0404	0.0193
	MAE	1.0714	0.2982	0.1653	0.2138	0.1100	0.0652	0.0941	0.2107	0.3288	0.0927	0.1105	0.0994
Macro-Economy	MSE	0.7312	0.7519	0.7086	0.7142	0.7104	0.7845	0.7092	0.7123	0.7224	0.7109	0.7022	0.7136
	MAE	0.6537	0.6728	0.6286	0.6307	0.6291	0.7106	0.6317	0.6337	0.6481	0.6321	0.6260	0.6308
SIR-Model	MSE	0.0198	0.0030	0.0019	0.0019	0.0026	0.5375	0.0020	0.0019	0.0019	0.0019	0.0034	0.0022
	MAE	0.1156	0.0407	0.0328	0.0328	0.0408	0.6662	0.0342	0.0333	0.0330	0.0334	0.0464	0.0361
Supply-Demand-Price	MSE	5.1740	0.1594	0.0601	0.1346	5.3351	1.6242	0.1181	0.1364	0.5075	0.2176	0.2048	0.3853
	MAE	1.6923	0.2762	0.1573	0.2607	1.5233	0.9403	0.2554	0.2567	0.4820	0.3137	0.3288	0.3986
Weather-Sales	MSE	0.8457	0.5425	0.6510	0.7205	0.6514	0.5084	0.5678	0.6791	0.7643	0.5285	0.5203	0.5476
	MAE	0.7195	0.5301	0.5802	0.6260	0.5805	0.5094	0.5547	0.5998	0.6514	0.5243	0.5211	0.5294

To evaluate the capacity of time series models to handle complex temporal patterns found in real-world applications, we generated synthetic datasets that simulate various real-world phenomena while maintaining controlled generation processes. These datasets include simulations of economic indicators, electricity consumption patterns, industrial sensor measurements, network traffic, stock prices, temperature readings, and website traffic for univariate testing, as well as advertising-sales

relationships, intervention effects, Lotka-Volterra dynamics, macroeconomic variables, epidemiological models, supply-demand-price relationships, and weather-sales correlations for multivariate scenarios.

As shown in Table 2, filter-based models excel with sensor-related data, with PaiFilter achieving best performance on Industrial-Sensor-Normal and Temperature-Sensor datasets. TimeKAN demonstrates superior performance with economic and financial time series, leading in Stock-Price and Retail-Sales forecasting. The results in Table 3 reveal that channel-dependent models (such as TSMixer, TimesNet, and TimeMixer) generally perform well on complex multivariate datasets, which aligns with our previous findings from the cross-variable learning experiments. This suggests that models designed to leverage relationships between variables have an inherent advantage when handling multivariate time series with complex interdependencies and feedback mechanisms. A particularly notable observation is that channel-dependent models do not consistently outperform channel-independent models across all multivariate datasets. For instance, while TSMixer (channel-dependent) excels on Weather-Sales and Lotka-Volterra datasets, PaiFilter (channel-independent) achieves the best performance on Supply-Demand-Price and SIR-Model datasets. This suggests that when variable relationships become more complex and nuanced, channel-dependent models may sometimes struggle to effectively learn these patterns, occasionally performing worse than channel-independent models that treat each variable separately. These results also highlight that no single architecture dominates across all patterns, with performance advantages highly dependent on the specific temporal characteristics of the target domain.

### B.3 Evaluating Zero-Shot Capabilities of Time Series Foundation Models

Table 4: Zero-Shot Forecasting Results by Pattern Type.

Pattern Type	Metric	Model		
		Chronos	Moirai	TimeMoE
Trend Patterns	MSE	<b>0.0002</b>	<u>0.9332</u>	30.9859
	MAE	<b>0.0047</b>	<u>0.3306</u>	1.5138
Periodic Patterns	MSE	<u>0.5293</u>	1.3476	<b>0.0594</b>
	MAE	<u>0.4885</u>	0.9125	<b>0.1353</b>
Complex Univariate	MSE	<b>0.4137</b>	0.6437	<u>0.5269</u>
	MAE	<b>0.3925</b>	0.5707	<u>0.4716</u>
Complex Multivariate	MSE	<b>0.7800</b>	<u>1.8564</u>	2.0845
	MAE	<b>0.5066</b>	<u>0.7266</u>	0.7574

Our evaluation of zero-shot forecasting capabilities reveals distinct pattern specialization among time series foundation models. As shown in Figure 3 and Table 4, each model demonstrates unique strengths aligned with specific temporal patterns.

Chronos excels at trend forecasting with near-perfect accuracy (MSE: 0.0002), capturing growth patterns without fine-tuning. Conversely, TimeMoE demonstrates exceptional capabilities in modeling periodic patterns (MSE: 0.0594), precisely predicting oscillatory behaviors while struggling significantly with trends. Moirai shows moderate but consistent performance across pattern types without particular specialization.

For complex datasets containing mixed patterns, Chronos maintains the best overall performance on both univariate and multivariate data, showing that Chronos’ approach of using Gaussian processes to generate synthetic data effectively enhances its generalization capabilities.

### B.4 Limitations of Normalization Layer

To investigate how different architectures handle long-range dependencies and trend transitions, particularly the impact of normalization layers, we designed two challenging synthetic time series patterns, as shown in Figure 4. These patterns were specifically chosen because their numerical values carry critical predictive information—a characteristic that could expose potential limitations of normalization techniques. The first is a triangle wave with an extended period of 400 time steps, where reaching values of 1 or -1 indicates an imminent trend reversal. The second is a pulse pattern

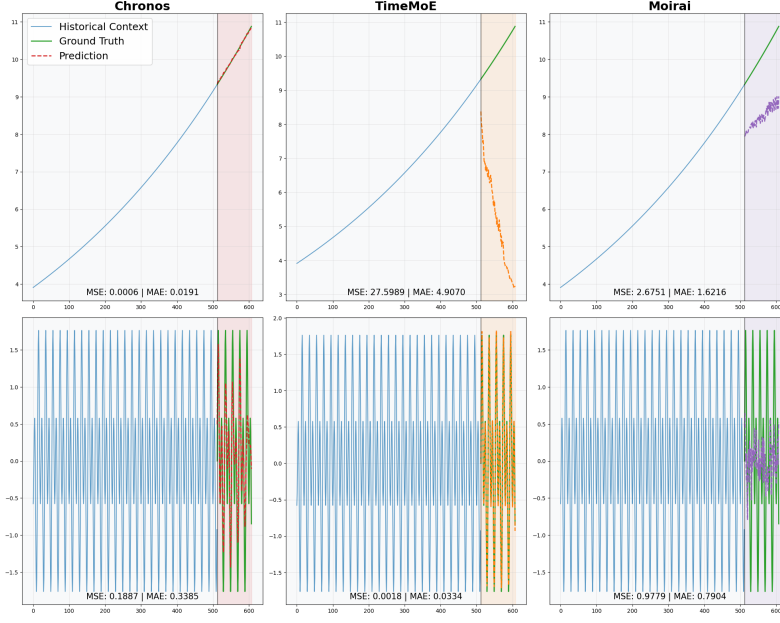


Figure 3: Zero-shot forecasting comparison between Chronos, TimeMoE, and Moirai models on different data types. The top row shows trend forecasting performance, while the bottom row displays periodic pattern forecasting capabilities.

96 with spikes appearing at regular intervals of 96 time steps, requiring models to accurately predict  
 97 both the timing and amplitude of subsequent pulses.

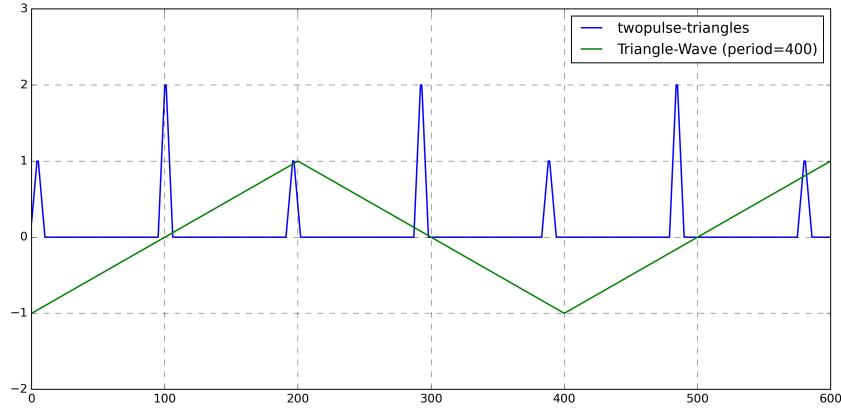


Figure 4: Synthetic time series patterns designed to test long-range dependencies: a triangle wave with period 400 (green) and a sequence with alternating pulses at regular intervals of 96 time steps (blue).

98 For both patterns, we used an input window of 96 time steps and asked models to predict the next  
 99 96 time steps. This experimental setup poses two specific challenges:

- 100 1. **Long-range dependency recognition:** For the pulse pattern, models must recognize that  
 101 after observing one pulse, another will appear exactly 96 steps later, potentially with a  
 102 different amplitude.
- 103 2. **Trend reversal prediction:** For the triangle wave, models must predict when the upward  
 104 or downward trend will reverse, despite having seen only a fraction of the full 400-step  
 105 cycle. This reversal is signaled by the series approaching its extreme values of 1 or -1.



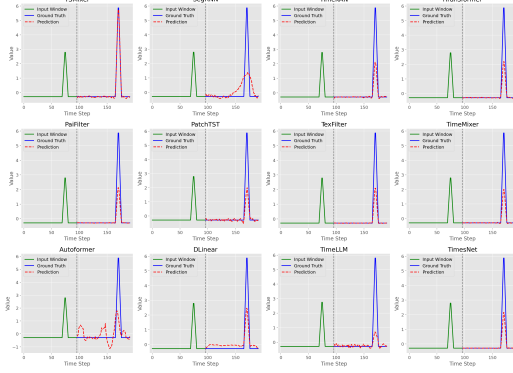


Figure 5: Model predictions on the pulse pattern. Each subplot shows a model’s prediction (red dashed line) compared to ground truth (blue solid line) and input window (green solid line). TSMixer accurately predicts the correct pulse height, while most batch-normalized models struggle to preserve magnitude information.

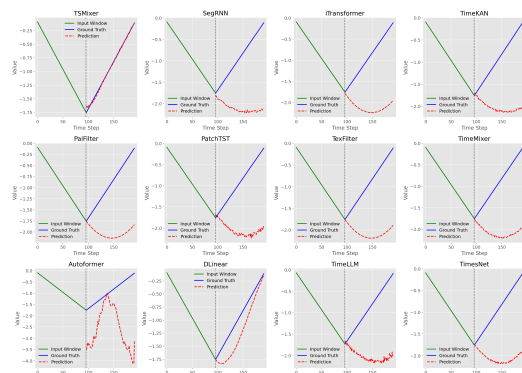


Figure 6: Model predictions on the triangle wave pattern. Each subplot shows a model’s prediction (red dashed line) compared to ground truth (blue solid line) and input window (green solid line). Models with batch normalization struggle to predict the correct trend reversal point.

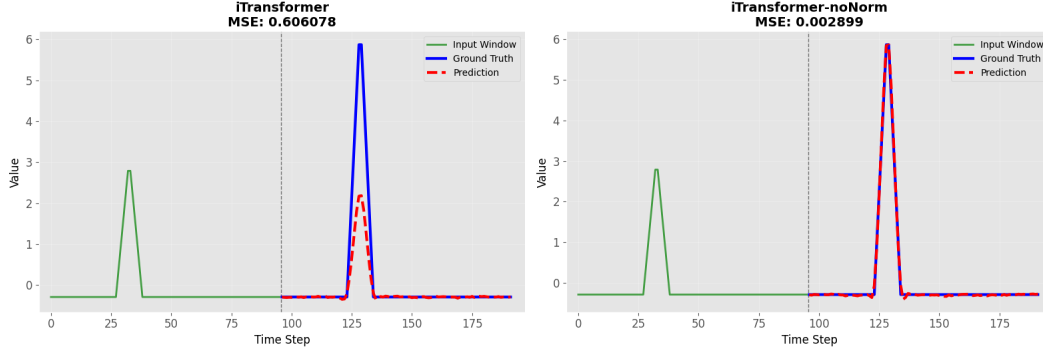


Figure 7: Left: iTransformer with normalization layers. Right: iTransformer-noNorm without normalization layers.

106 Our experimental results, visualized in Figures 5 and 6, reveal a striking pattern: among all tested  
 107 models, only TSMixer demonstrates superior performance. Notably, of all models tested, only Seg-  
 108 RNN, DLinear, and TSMixer do not employ normalization layers, with TSMixer showing the most  
 109 consistent accuracy.

110 In the pulse pattern forecasts (Figure 5), TSMixer correctly predicts both the timing and magnitude  
 111 of the upcoming pulse. In contrast, models with normalization layers like Autoformer show erratic  
 112 behavior, while others like PaiFilter, TexFilter, and TimeMixer predict pulses with significantly re-  
 113 duced amplitudes. This suggests that normalization is stripping away critical amplitude information.

114 For the triangle wave pattern (Figure 6), the impact of normalization is even more evident. TSMixer  
 115 accurately follows the downward trend and correctly predicts the exact point where the trend re-  
 116 verses, perfectly aligning with the ground truth throughout the entire prediction window. Models  
 117 using normalization layers largely fail to identify the correct trend reversal point, with most predict-  
 118 ing either a continued downward trajectory or an incorrect reversal pattern.

119 To further validate our hypothesis about normalization layers, we conducted a comparative exper-  
 120 iment with iTransformer, creating a variant (iTransformer-noNorm) with all normalization layers  
 121 removed. As shown in Figure 7, the normalization-free variant dramatically outperforms the origi-  
 122 nal model on these patterns, confirming that normalization layers are indeed the limiting factor.

123 These findings highlight a fundamental limitation of normalization in time series forecasting: the  
 124 normalization process inherently discards critical magnitude information. When normalization stan-

125 dardizes features to have zero mean and unit variance, it effectively removes the absolute scale infor-  
 126 mation that is often crucial for time series patterns. While this standardization is beneficial for  
 127 stable training and handling distribution shifts in many deep learning applications, it proves detri-  
 128 mental when absolute magnitude carries important predictive information.

## 129 B.5 Experimental Supplementary Content

130 This section provides additional experimental results and visualizations that complement the main  
 131 findings presented in the paper. We include detailed model performance comparisons across differ-  
 132 ent pattern types, cross-variable relationship analysis, and comprehensive anomaly impact assess-  
 133 ment.

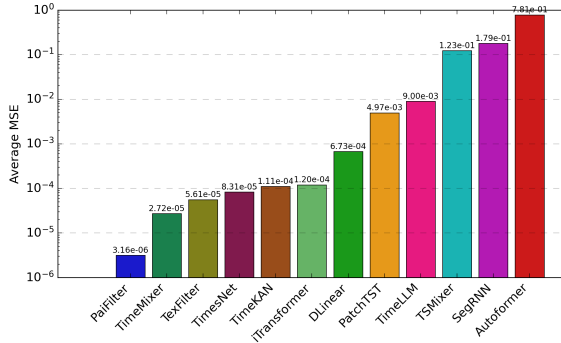


Figure 8: Average MSE comparison for different models on trend data.

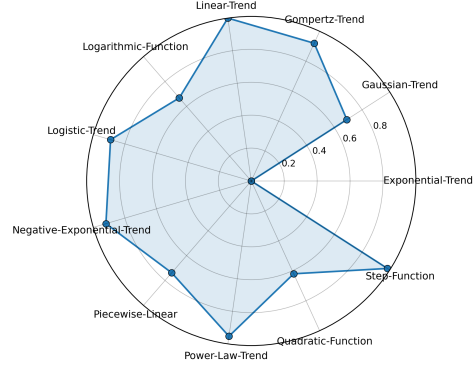


Figure 9: Radar chart visualizing the prediction difficulty of different trend patterns. Lower values indicate more challenging patterns to predict.

134 Figure 8 illustrates model performance on trend data, where PaiFilter consistently achieves the  
 135 lowest error rates, demonstrating superior capability in capturing trend relationships compared to  
 136 other architectures. Figure 9 presents a radar chart of trend pattern difficulty. To construct this  
 137 visualization, we first calculated the average MSE for each signal type across all models, then ap-  
 138 plied logarithmic transformation ( $\log_{10}$ ) to compress the range of values. The transformed val-  
 139 ues were then normalized to a 0-1 scale using the formula:  $normalized\_value = 1 - (val - min\_val) / (max\_val - min\_val)$ , where lower values in the radar chart indicate more chal-  
 140 lenging patterns. This inverse mapping ensures that patterns with higher average prediction error (more  
 141 difficult) appear closer to the center of the radar chart. The visualization reveals that patterns with  
 142 accelerating growth rates, specifically Quadratic-Function and Exponential-Trend, pose the greatest  
 143 challenge for forecasting models, as their rapid rate changes become increasingly difficult to predict  
 144 accurately over longer horizons.

146 Figure 10 presents model performance on periodic data, where TimesNet and DLinear demonstrate  
 147 exceptional capabilities in modeling oscillatory patterns. The difficulty radar chart in Figure 11 was  
 148 calculated using the same methodology as for trend patterns: first computing average MSE across all  
 149 models for each pattern type, applying logarithmic transformation to compress the range, and finally  
 150 normalizing and inverting the values to create the 0-1 scale where lower values indicate more chal-  
 151 lenging patterns. The radar chart clearly indicates that complex multi-frequency patterns (Ten-Sin)  
 152 and discontinuous patterns (Square-Wave) present the greatest challenges. Square waves, with their  
 153 abrupt value transitions, are particularly difficult for models to predict accurately, as most architec-  
 154 tures struggle to capture these discontinuities without introducing smoothing effects or oscillatory  
 155 artifacts.

156 Figure 12 illustrates model performance in detecting temporal lag relationships, with both input  
 157 window and forecasting horizon set to 96 time steps. The experiment involved predicting two vari-  
 158 ables: var1 (random noise) and var2 (var1 lagged by different time steps). As the lag value increases  
 159 from 5 to 48, channel-dependent architectures (TimesNet, TSMixer, TimeMixer, and iTransformer)  
 160 show progressively decreasing MSE values for var2, indicating these models successfully capture

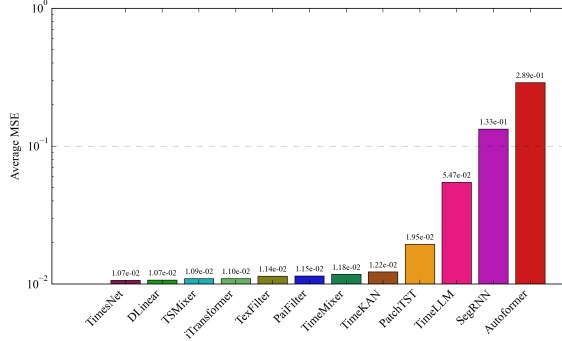


Figure 10: Average MSE comparison for different models on periodic data.

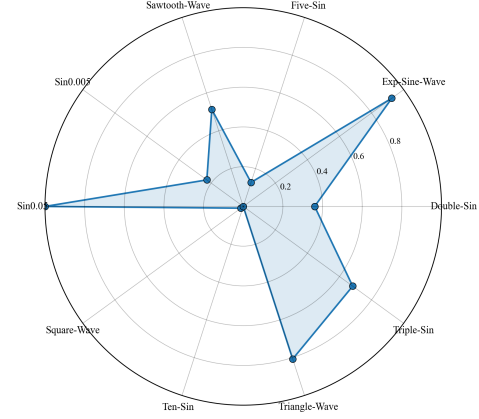


Figure 11: Radar chart visualizing the prediction difficulty of different periodic patterns. Lower values indicate more challenging patterns to predict.

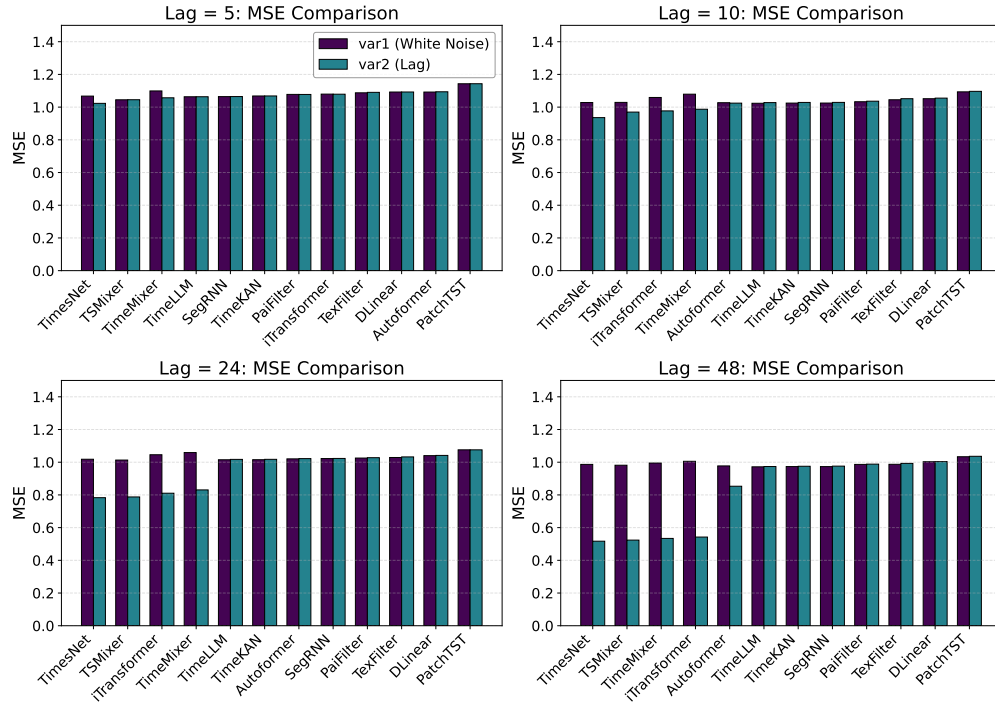


Figure 12: MSE comparison for lag relationship detection with different lag values (5, 10, 24, and 48). Blue bars represent MSE for white noise prediction (var1), while red bars represent MSE for the lagged variable prediction (var2). Lower MSE for the lagged variable indicates better ability to detect temporal relationships.

161 the temporal relationships despite increasing lag distances. This capability reveals their effective-  
 162 ness in leveraging cross-channel information to identify and model temporal dependencies between  
 163 variables.

164 Figure 13 evaluates models on the Sine-Noise dataset, which comprises three variables: var1 (sine  
 165 wave), var2 (white noise), and var3 (sum of var1 and var2). This dataset is specifically designed to  
 166 test models' ability to learn relationships between variables. If models predict var3 independently  
 167 (treating each channel separately), they will be affected by the noise component from var2, resulting

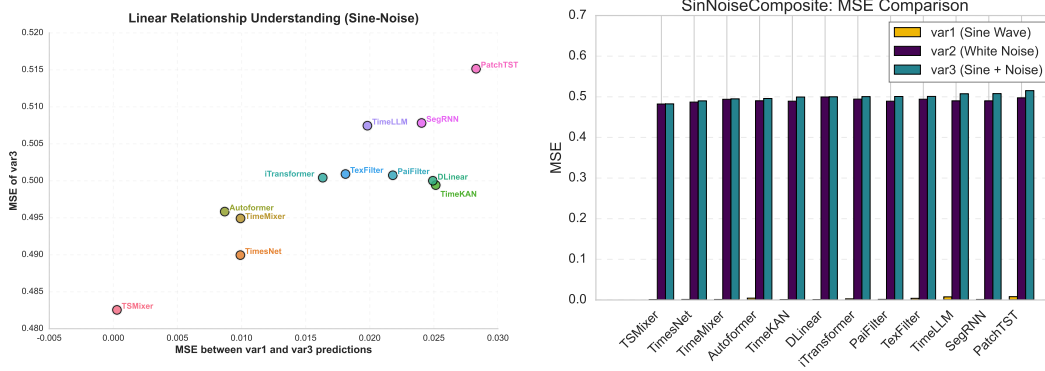


Figure 13: Test results on the Sine-Noise dataset for evaluating models’ ability to capture relationships between variables. This dataset consists of var1 (sine wave), var2 (white noise), and var3 (sum of var1 and var2). Left: Scatter plot showing prediction performance for understanding linear relationships, where x-axis represents MSE between var1 and var3 predictions, while y-axis shows MSE of var3 predictions vs ground truth. Models in the bottom-left corner better understand the additive relationship between variables. Right: Per-model MSE values for the three variables, where lower values for var1 and var3 compared to var2 indicate the model effectively captures the relationship between the sine component and the composite signal.

in higher prediction errors. However, if models successfully learn the additive relationship  $\text{var3} = \text{var1} + \text{var2}$ , they can recognize that the true signal component of var3 is simply var1, since var2 is unpredictable noise. In this case, models should achieve low MSE for var3 predictions by effectively filtering out the noise. Since var2 is white noise, models typically predict its value as 0, and consequently, if the relationship is properly learned, the predicted values for var3 should closely match those for var1. The left scatter plot reveals a positive correlation between MSE of var3 predictions and MSE between predicted var1 and var3. Notably, TSMixer achieves near-zero MSE between its var1 and var3 predictions, resulting in the lowest overall var3 prediction error. The right bar chart confirms TSMixer’s superior performance across all three variables, demonstrating its exceptional ability to capture and leverage the additive relationship between variables.

Table 5: Comprehensive Comparison of MSE and MAE for Different Anomaly Types

Anomaly Type	Metric	Model										
		Autoformer	DLinear	PaiFilter	PatchTST	SegRNN	TSMixer	TexFilter	TimeKAN	TimeMixer	TimesNet	iTransformer
No Anomaly	MSE	0.0935	0.0390	<u>0.0205</u>	0.0257	0.0863	0.0277	0.0207	0.0207	0.0206	<b>0.0203</b>	0.0209
	MAE	0.1871	0.1392	<b>0.1086</b>	0.1260	0.1866	0.1227	0.1091	0.1097	0.1094	<u>0.1087</u>	0.1098
point-anomaly-0.5pct	MSE	0.1619	0.0280	0.0246	0.0300	0.0914	0.1344	0.0238	<b>0.0204</b>	<b>0.0204</b>	0.0238	0.0254
	MAE	0.2544	0.1254	0.1197	0.1361	0.1919	0.2700	0.1157	<u>0.1079</u>	<b>0.1078</b>	0.1174	0.1216
point-anomaly-1pct	MSE	0.1795	0.0209	0.0210	0.0233	0.0841	0.0663	0.0222	<u>0.0186</u>	<b>0.0175</b>	0.0220	0.0223
	MAE	0.2681	0.1073	0.1092	0.1185	0.1810	0.1873	0.1142	<u>0.1041</u>	<b>0.1007</b>	0.1136	0.1150
point-anomaly-5pct	MSE	0.1090	0.0198	0.0214	0.0304	0.0766	0.1519	0.0309	<b>0.0143</b>	<u>0.0193</u>	0.0322	0.0293
	MAE	0.2221	0.1053	0.1078	0.1329	0.1803	0.2805	0.1345	<b>0.0846</b>	<u>0.1024</u>	0.1390	0.1301
point-anomaly-10pct	MSE	0.0726	0.0579	0.0209	0.0280	0.0669	0.2046	0.0314	<b>0.0163</b>	<u>0.0206</u>	0.0542	0.0361
	MAE	0.1897	0.1419	0.1064	0.1265	0.1787	0.2833	0.1351	<b>0.0950</b>	<u>0.1060</u>	0.1686	0.1436
pulse-anomaly-1	MSE	0.2019	0.0625	0.0240	0.0303	0.0991	0.0540	0.0212	<u>0.0211</u>	0.0260	<b>0.0198</b>	0.0219
	MAE	0.2775	0.1872	0.1109	0.1306	0.1986	0.1673	0.1107	<u>0.1093</u>	0.1143	<b>0.1067</b>	0.1125
pulse-anomaly-3	MSE	0.1271	0.1513	<u>0.0226</u>	0.0381	0.1078	0.2714	<b>0.0213</b>	0.0358	0.0255	0.0285	0.0387
	MAE	0.2371	0.2501	<b>0.0926</b>	0.1139	0.2062	0.3640	<u>0.0931</u>	0.0945	0.0959	0.0966	0.1068
pulse-anomaly-5	MSE	0.1271	0.1169	<u>0.0145</u>	0.0218	0.0994	0.1764	0.0221	<b>0.0137</b>	0.0160	0.0180	0.0189
	MAE	0.2266	0.2655	<u>0.0950</u>	0.1158	0.2066	0.2968	0.1085	<b>0.0921</b>	0.0989	0.1065	0.1071

Table 5 presents model performance under various anomaly conditions, including point anomalies (isolated outliers) at different densities (0.5% to 10%) and pulse anomalies (sustained deviations) with varying frequencies (1, 3, or 5 pulses). The results demonstrate significant variance in model robustness against data irregularities. TimeKAN and TimeMixer exhibit strong resilience against point anomalies, maintaining consistent performance as anomaly density increases. For pulse anomalies, TimesNet and TexFilter perform best with fewer pulses, while PaiFilter and TimeKAN demonstrate superior handling of cases with multiple pulses. Notably, TSMixer shows considerable performance degradation as anomaly density increases, highlighting a vulnerability that could be critical in real-world applications where data quality is variable.

## C Detailed Introduction of the Generated Dataset

This section provides a comprehensive overview of the synthetic time series datasets generated within our SynTSBench framework. We systematically introduce the building blocks and methodology behind our dataset generation process, which forms the foundation for rigorous time series forecasting evaluation. The section is organized into five key components: Appendix C.1 presents the fundamental signal types that serve as building blocks for more complex patterns; Appendix C.2 describes how we systematically introduce controlled noise and anomalies to test model robustness; Appendix C.3 explains our approach to evaluating models across varying temporal scales; Appendix C.4 details the generation of complex univariate time series that simulate real-world phenomena; and Appendix C.5 introduces our multivariate datasets with explicit causal relationships and interdependencies. Together, these components enable comprehensive evaluation of time series forecasting models across diverse temporal patterns, noise conditions, and complexity levels.

### C.1 Basic Signals

To establish a foundation for systematically evaluating time series forecasting models, we generate a comprehensive set of synthetic signals that simulate real-world temporal patterns. Table 6 presents the basic building blocks used in our framework **SynTSBench**, categorized into three fundamental classes: trend functions, periodic functions, and other signal types. Trend functions simulate directional movements commonly observed in economic growth, population changes, and technology adoption curves. Periodic functions represent cyclical behaviors found in seasonal fluctuations, biological rhythms, and industrial processes. The third category encompasses stochastic processes and composite signals that capture the complexity of financial markets, sensor readings, and natural phenomena. By generating controlled combinations of these signals with programmable parameters, we create synthetic datasets that isolate specific temporal features, enabling us to precisely map model capabilities to pattern types and establish theoretical performance boundaries for rigorous evaluation.

### C.2 Signals with Noise and Anomalies

To systematically evaluate model robustness against common data irregularities, we generated synthetic datasets incorporating controlled noise and anomalies. We selected representative signals from both trend patterns (linear trend, logistic trend) and periodic patterns (sin0.05, Double-Sin, five-Sin) as base signals, and applied varying levels of Gaussian noise (30dB, 20dB, 10dB, 0dB, -10dB, and no noise). This approach enables quantitative assessment of model performance degradation across the noise spectrum.

For anomaly testing, we injected two types of irregularities—point anomalies (random spikes) at varying densities (1%, 5%, 10%) and pulse anomalies (sustained deviations with 1, 3, or 5 pulses)—on top of signals with 20dB noise. Importantly, anomalies were only introduced in the first 80% of each time series, keeping the test portion (last 20%) anomaly-free. This design enables direct comparison of forecasting performance across different anomaly severities, revealing each model’s sensitivity to historical anomalies and recovery capabilities. For visualization examples of these synthetic datasets, refer to Figure 14 for noise levels and Figure 15 for anomaly types.

### C.3 Datasets with Different Length

For evaluating model performance across varying time series lengths, we generated synthetic datasets with lengths of 1000, 5000, 10000, and 20000 time steps. We selected six representative signal patterns: three trend functions (“Linear-Trend”, “Quadratic-Function”, “Exponential-Trend”) and three periodic functions (“Sin0.005”, “Double-Sin”, “Five-Sin”). Each signal type was generated with four distinct noise levels (SNR of 20dB, 0dB, -10dB, and no noise) to provide a comprehensive evaluation framework. This dataset design enables quantitative analysis of each model’s ability to handle increasing historical context and their performance scaling properties across different temporal patterns and data volumes.

Table 6: Trend, Periodic, and Other Signals for Time Series Analysis

Function Name	Mathematical Formula	Brief Description
<b>Trend Functions</b>		
Linear	$y = at + b$	Represents linear growth or decline over time $t$ .
Quadratic	$y = at^2 + bt + c$	Represents parabolic trend, indicating acceleration or deceleration.
Exponential	$y = ae^{bt}$	Represents rapid growth or decay over time $t$ .
Logarithmic	$y = a \ln(t) + b$	Represents initial rapid growth that slows over time.
Logistic	$y = \frac{L}{1 + e^{-k(t-t_0)}}$	Represents S-shaped growth, often used for resource-limited scenarios.
Gompertz	$y = ae^{-be^{-kt}}$	Represents growth that starts rapidly and slows, used in biology and economics.
Power Law	$y = at^b$	Represents scaling relationships or long-tailed distributions.
Step Function	$y = \begin{cases} c & t < t_0 \\ d & t \geq t_0 \end{cases}$	Represents a sudden change at a specific point in time.
Piecewise Linear	$y = \begin{cases} a_1t + b_1 & t < t_1 \\ a_2t + b_2 & t_1 \leq t < t_2 \\ \dots & \dots \end{cases}$	Represents different linear trends in different time intervals.
Gaussian	$y = ae^{-\frac{(t-t_0)^2}{2\sigma^2}}$	Represents a bell-shaped trend, useful for lifecycle analysis.
<b>Periodic Functions</b>		
Sine Wave	$y = A \sin(2\pi ft + \phi)$	Models smooth cyclic phenomena like ocean tides, seasonal temperature variations, or sound wave pressure.
Triangle Wave	$y = \frac{2A}{\pi} \arcsin(\sin(2\pi ft))$	Models linear charging/discharging cycles in electronics, or simplified vibrations in mechanical systems.
Square Wave	$y = A \cdot \text{sgn}(\sin(2\pi ft))$	Models on-off switching patterns like digital signals, heartbeats, machinery with two distinct states, or control systems with binary outputs.
Sawtooth Wave	$y = 2A(t/T - \lfloor t/T + 1/2 \rfloor)$	Models rapid return phenomena such as voltage in electrical circuits with capacitors, ramp generators, or instrument sounds like brass.
Composite Sine	$y = \sum_{i=1}^n A_i \sin(2\pi f_i t + \phi_i)$	Models complex periodic phenomena like musical tones with harmonics, combined seasonal effects, or multiple overlapping business cycles.
Exponential Sine	$y = e^{\sin(t)}$	Models amplitude-modulated oscillations in signals, or periodic systems with exponentially varying intensity.
<b>Other Signals</b>		
ARMA Signal	$x_t = \mu + \sum_{i=1}^p \phi_i x_{t-i} + \sum_{j=1}^q \theta_j \epsilon_{t-j} + \epsilon_t$	Models systems with memory and feedback, such as financial returns, temperature fluctuations, or industrial production.
Random Walk	$x_t = x_{t-1} + \epsilon_t$	Models unpredictable accumulating processes like stock prices, exchange rates, or particle movements.
White Noise	$x_t = \epsilon_t, \epsilon_t \sim N(0, \sigma^2)$	Models purely random fluctuations like measurement errors, static in communication, or thermal noise in electronics.
Composite Signal	$x_t = \sum_{i=1}^n \alpha_i x_i(t)$	Models real-world complex systems combining multiple patterns, such as economic indicators with trends, seasonality and noise.



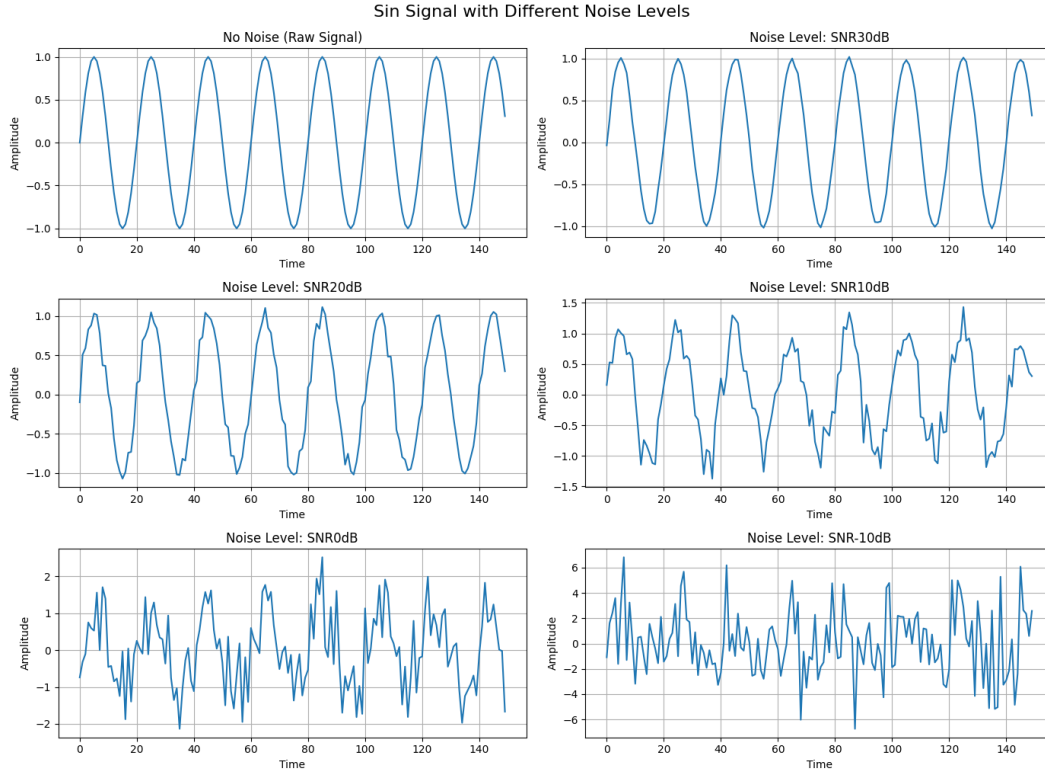


Figure 14: Sine wave signal with progressive noise levels. From top to bottom: clean signal, 30dB, 20dB, 10dB, 0dB, and -10dB SNR. Higher SNR values indicate less noise, while negative SNR values represent cases where noise dominates the signal.

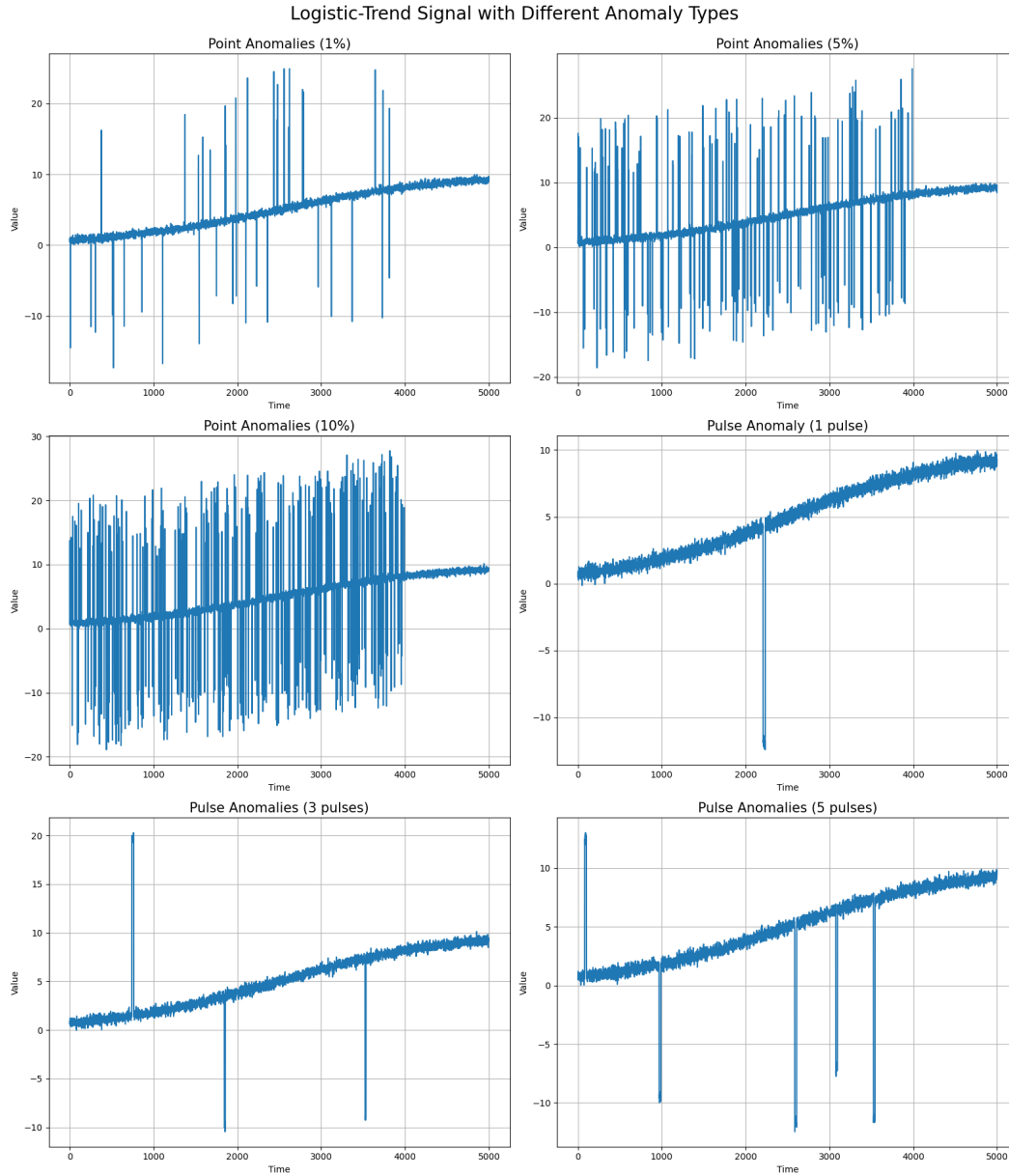


Figure 15: Logistic trend signals with various anomaly types. Top row shows point anomalies at 1% and 5% densities, middle row shows point anomalies at 10% and a single pulse anomaly, bottom row shows multiple pulse anomalies (3 and 5 pulses).

## C.4 Complex Univariate Datasets

To comprehensively evaluate model performance on complex time series patterns, we generated synthetic datasets that combine multiple basic signals to simulate complex temporal behaviors found in various domains. These composite signals capture key characteristics of real-world time series while maintaining fully controlled generation processes, enabling precise evaluation of model performance across diverse pattern types.

For example, our stock price simulator combines five essential components: (1) a slow exponential trend representing long-term market growth, (2) overlapping sine waves with different frequencies simulating market cycles, (3) higher-frequency seasonal patterns, (4) a random walk component capturing market noise according to the efficient market hypothesis, and (5) volatility clusters using GARCH-like behavior to model periods of market stress. Similarly, our temperature sensor simulation combines annual and daily periodic patterns with varying amplitudes, a gradual trend component, and realistic noise.

Other simulated patterns include electricity consumption with daily, weekly, and seasonal components; industrial sensor readings with degradation patterns; network traffic with diurnal and weekly patterns plus random bursts; retail sales with multiple seasonal effects; and economic indicators with trend-cycle-noise components. Each pattern type incorporates domain-specific characteristics essential for realistic forecasting challenges. Figure 16 provides visualization examples of these synthesized time series, demonstrating their diversity and complexity. The complete generation details are available in our open-source codebase.

## C.5 Complex Multivariate Datasets

We generated a diverse set of multivariate time series datasets to evaluate model performance on realistic and interdependent temporal patterns. These datasets include both classical dynamic systems and synthetic scenarios with explicit causal relationships. For example, we used differential equation models such as the Lotka-Volterra equations (predator-prey dynamics) and the SIR epidemic model to create time series where variables interact according to well-defined mathematical laws. Additionally, we constructed datasets with clear causal structures, such as the Weather-Sales dataset (where temperature and rainfall influence ice cream, umbrella, and beverage sales), as well as scenarios modeling advertising-sales effects, macroeconomic indicators, supply-demand-price interactions, and intervention effects. The relationships among variables in these datasets are designed to reflect real-world dependencies, such as weather driving consumer behavior or supply and demand jointly determining prices. Figures 17 to 19 illustrate some representative examples, while many more multivariate datasets and their generation details can be found in our open-source codebase.

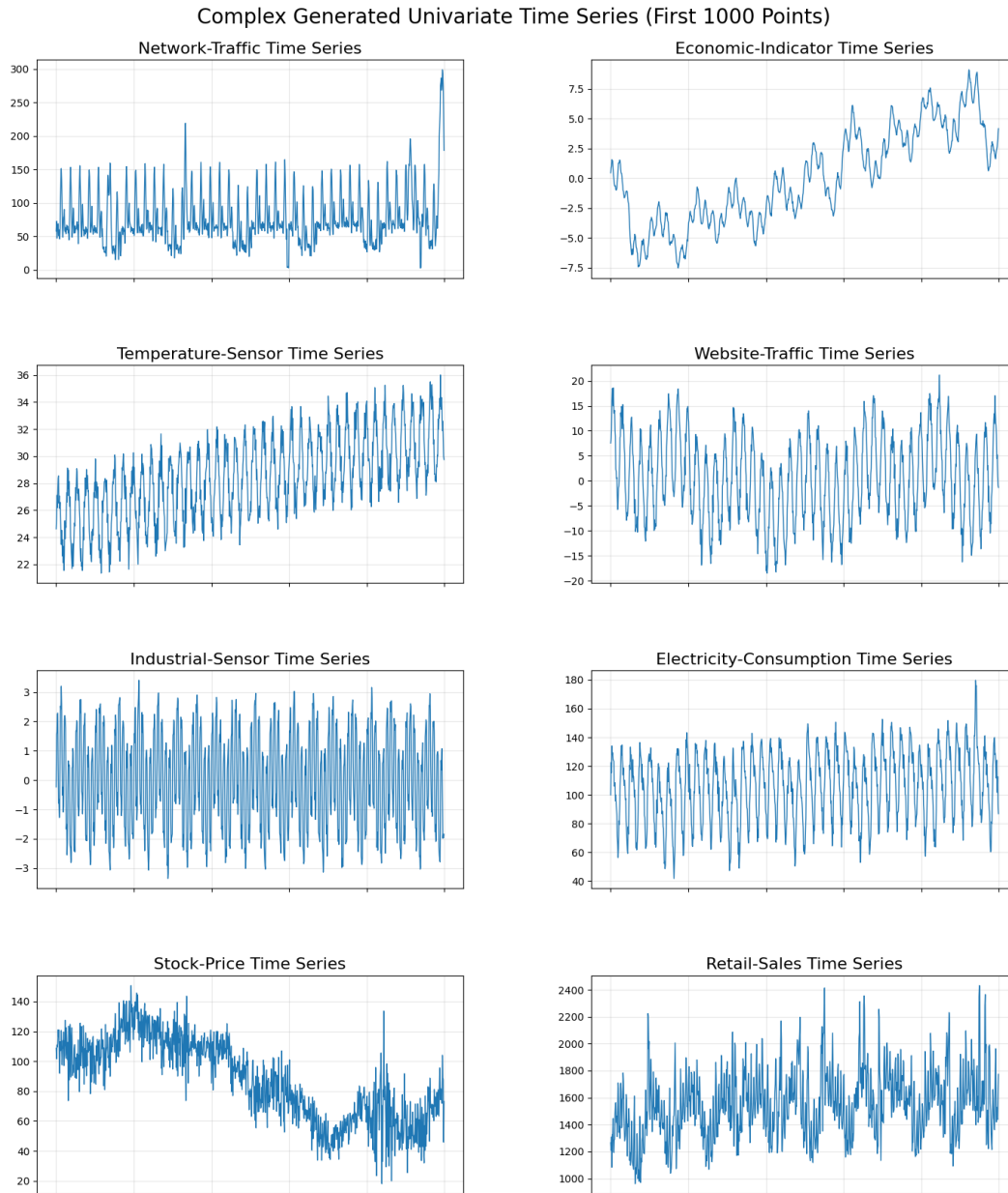


Figure 16: Examples of complex univariate time series generated by combining multiple basic signal patterns.

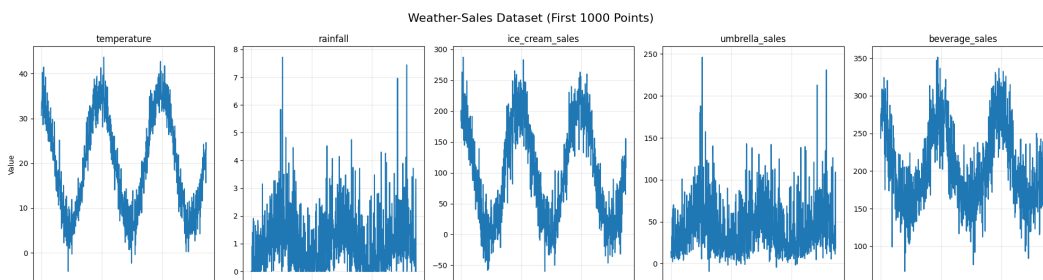


Figure 17: Weather-Sales dataset showing relationships between weather variables (temperature, rainfall) and consumer purchases (ice cream sales, umbrella sales, beverage sales).

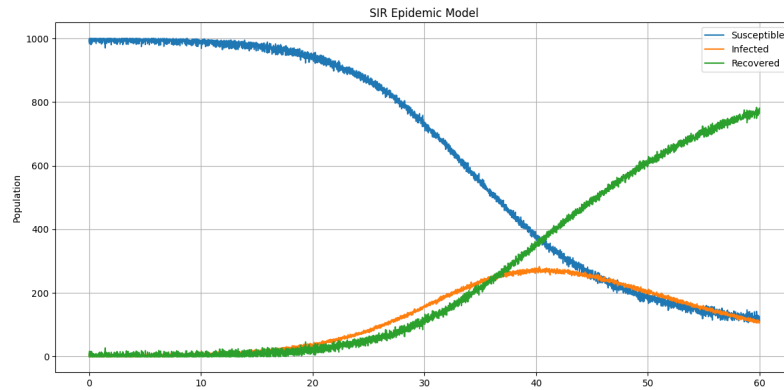


Figure 18: SIR epidemic model simulation showing the progression of an outbreak through Susceptible (blue), Infected (orange), and Recovered (green) populations over time.

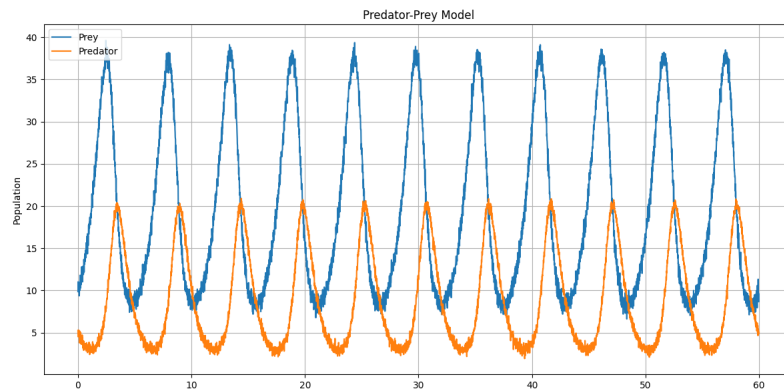


Figure 19: Lotka-Volterra predator-prey model demonstrating the cyclical relationship between prey (blue) and predator (orange) populations, where changes in one population affect the other with a time delay.

## D Experiment Details

### D.1 Experiment Parameters

Our experiments were conducted on high-performance computing infrastructure consisting of four NVIDIA A800-SXM GPUs, each with approximately 79.3 GB (81251 MiB) of memory, running CUDA version 12.4. For dataset partitioning, we employed different strategies based on model types: deep learning models used a 7:1:2 ratio for training, validation, and testing sets respectively, while traditional models used an 8:2 split for training and validation. All evaluations employed a sliding window prediction approach with a stride of 1, where traditional models utilized the entire historical data available prior to the prediction point.

Table 7 presents the detailed experimental configuration across different evaluation scenarios, including input window sizes, forecast horizons, and dataset lengths used for each experiment type. Table 8 summarizes the diverse set of forecasting models evaluated in our benchmark, spanning multiple architectural paradigms from transformers to linear models. Table 9 details the hyperparameter configurations for each full-shot model. For zero-shot forecasting models, we used the following model variants: chronos-bolt-base, TimeMoE-200M, and Moirai-small. These models were loaded using the official implementations from their respective GitHub repositories: Chronos (<https://github.com/amazon-science/chronos-forecasting>), TimeMoE (<https://github.com/Time-MoE/Time-MoE>), and Moirai (<https://github.com/SalesforceAIResearch/uni2ts>).

Table 7: Experiment Parameters for Different Dataset Parts. *Note:* “Original Scale Evaluation” indicates whether evaluation metrics were calculated after inverting normalized values back to their original scale (Yes) or directly on the normalized values (No).

Experiment Type	Input Window	Forecast Horizons	Dataset Length	Original Scale Evaluation	Running Time (hours)
Temporal Pattern Learning (Trend Period Testing)	96	24, 48, 96, 192	5000	No	30
Robustness Against Noise & Anomalies	96	96	5000	No	20
Short/Long-range Dependencies	96	10	5000	Yes	50
Cross-Variable Learning	96	96	5000	No	3
Dataset Scale Sensitivity	96	96	1000, 5000, 10000, 20000	No	40
Complex Real-World Pattern Simulation	96	96	5000	No	10
Zero-shot Model Testing	512	96	5000	No	3

Table 8: Model Overview Table

Model Name	Framework Type	Key Features
PatchTST [28]	Transformer	Long sequence forecasting, sub-sequence input
TimeLLM [50]	Transformer	Based on LLM, textual original input
Autoformer [27]	Transformer	Decomposes Transformer, self-adaptive mechanism
iTransformer [31]	Transformer	Inverted Transformer, improved forecasting
TimeMixer [33]	MLP	Multi-dimensional fusion, high efficiency
TSMixer [32]	MLP	Fully MLP-based, scalable forecasting
TimesNet [25]	CNN	2D convolution kernel, periodic/trend learning
SegRNN [21]	RNN	Segment-based RNN, long sequence forecasting
DLinear [9]	Linear Model	Simple linear baseline
TimeKAN [51]	KAN	Interpretable, parameterized KAN
Paifilter [34]	FilterNet+MLP	Frequency-domain filtering
Texfilter [34]	FilterNet+MLP	Frequency-domain transformation
Chronos [35]	Transformer	Tokenizes series, probabilistic sampling
TimeMoE [41]	Transformer (MoE)	Sparse mixture-of-experts, billion-scale
Moirai [39]	Masked Encoder Transformer	Masked-encoder pretraining, multi-patch

Table 9: Hyperparameter Settings for Time Series Models

Model	Architecture Parameters	d_model	d_ff	learning_rate	training_epoch	Other Parameters
TimeKAN	e.layers=2	16	32	0.01	30	down_sampling_layers=2, down_sampling_window=2, begin_order=0
TimeMixer	e.layers=2	16	32	0.01	30	down_sampling_layers=2, down_sampling_method="avg", down_sampling_window=2
TimesNet	e.layers=2	16	32	0.001	30	top_k=5, num_kernels=6
TSMixer	e.layers=2	512	2048	0.001	30	factor=3
SegRNN	seg_len=2	512	-	0.0001	30	-
PatchTST	e.layers=3	512	2048	0.0001	30	patch_len=16, n_heads=4, factor=3
iTransformer	e.layers=3	512	512	0.0005	30	n_heads=8, factor=3
DLinear	-	-	-	0.01	30	label_len=10, moving_avg=25
Autoformer	e.layers=2, d.layers=1	512	2048	0.001	30	label_len=10, factor=3, moving_avg=25, n_heads=8
Paifilter	-	-	-	0.01	30	hidden_size=256
TexFilter	-	-	-	0.001	30	embed_size=512, hidden_size=512
TimeLLM	llm_layers=32	32	128	0.01	5	patch_len=16, stride=8, label_len=10, factor=3, n_heads=8



## 286 D.2 Evaluation Framework

### 287 D.2.1 Evaluation Metrics

288 **Mean Absolute Error (MAE)** The Mean Absolute Error (MAE) quantifies the average absolute  
289 deviation between the predicted values and the actual values. It treats all errors with equal weight  
290 regardless of their magnitude, making it robust to outliers. MAE is particularly useful when the cost  
291 of errors increases linearly with their size. Its mathematical formula is given by:

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t| \quad (1)$$

292 where  $y_t$  represents the actual value at time  $t$ ,  $\hat{y}_t$  represents the predicted value at time  $t$ , and  $n$  is  
293 the total number of observations.

294 **Mean Squared Error (MSE)** The Mean Squared Error (MSE) measures the average of the squares  
295 of the errors between predicted and actual values. By squaring the errors before averaging, MSE  
296 penalizes larger errors more heavily than smaller ones, making it especially sensitive to outliers. It  
297 is widely used when large errors are particularly undesirable. The mathematical representation is:

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2 \quad (2)$$

298 **Root Mean Squared Error (RMSE)** The Root Mean Squared Error (RMSE) is the square root of  
299 MSE, bringing the error metric back to the same unit as the original data. This makes RMSE more  
300 interpretable than MSE while still maintaining the property of penalizing large errors more heavily.  
301 RMSE is commonly used in regression problems and time series forecasting. It can be calculated  
302 as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2} \quad (3)$$

303 **Mean Absolute Percentage Error (MAPE)** The Mean Absolute Percentage Error (MAPE) ex-  
304 presses the prediction error as a percentage of the actual value, providing a scale-independent mea-  
305 sure of accuracy. This makes it useful for comparing forecast performance across different datasets  
306 with varying scales. However, MAPE is undefined for zero values and can give misleadingly high  
307 errors for small actual values. Its formula is:

$$\text{MAPE} = \frac{1}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right| \times 100\% \quad (4)$$

308 **Symmetric Mean Absolute Percentage Error (SMAPE)** The Symmetric Mean Absolute Percent-  
309 age Error (SMAPE) is a modified version of MAPE that addresses some of its limitations. By using  
310 the average of actual and predicted values in the denominator, SMAPE avoids division by zero and  
311 provides a more balanced measure that's bounded between 0% and 100%. It treats positive and  
312 negative errors symmetrically, making it suitable for evaluations where both over-forecasting and  
313 under-forecasting are equally important. The formula is:

$$\text{SMAPE} = \frac{100\%}{n} \sum_{t=1}^n \frac{|y_t - \hat{y}_t|}{\frac{|y_t| + |\hat{y}_t|}{2}} \quad (5)$$

### 314 D.2.2 Theoretical Performance Boundaries

315 One significant advantage of using synthetic datasets is that we can precisely determine the theoret-  
316 ical optimal prediction for each time series type. In real-world datasets, the underlying data gener-  
317 ation process is typically unknown, making it impossible to establish true performance boundaries.

318 With our synthetic approach, we can separate each time series into its predictable and unpredictable  
319 components, allowing us to calculate the theoretical minimum error achievable by any forecasting  
320 model.

321 **Trend and Periodic Datasets** For trend and periodic datasets without noise, the time series are  
322 generated by deterministic mathematical functions without any stochastic components. Since these  
323 patterns are entirely predictable given sufficient historical data, the theoretical optimal MSE and  
324 MAE values are exactly zero. This provides a clear benchmark against which to measure model  
325 performance on pure pattern learning.

326 **Stochastic Processes** For stochastic processes, the optimal prediction strategy depends on the  
327 specific process characteristics:

- 328 • For random walk processes, the optimal prediction is the naive forecast (i.e.,  $\hat{y}_{t+1} = y_t$ ),  
329 since by definition  $y_{t+1} = y_t + \epsilon_t$  where  $\epsilon_t$  is unpredictable noise. Therefore, the optimal  
330 MSE/MAE is calculated using the naive forecasting method.
- 331 • For white noise processes, the optimal prediction is the mean of the process (assuming prior  
332 knowledge that the series is white noise but without knowing its mean), as each observation  
333 is independent of previous values.
- 334 • For ARMA processes, the optimal prediction comes from the correctly specified ARMA  
335 model with the true parameter values used in data generation, as this captures all predictable  
336 temporal dependencies.

337 **Datasets with Noise** For datasets with added noise, we know both the observed values and the  
338 true underlying signal. The optimal prediction is based on the true signal component without noise.  
339 The optimal MSE/MAE is calculated by comparing this true signal with the observed values, repre-  
340 senting the theoretical minimum error achievable in the presence of observation noise.

341 **Complex Time Series Datasets** For complex time series like Network-Traffic or Stock-Price that  
342 combine multiple components (trend, seasonality, event-driven spikes, and noise), we identify which  
343 components are theoretically predictable from historical data (typically trend and seasonality) and  
344 which are not (random events and noise). The optimal prediction includes only the predictable  
345 components, and the optimal MSE/MAE is calculated by comparing these predictable components  
346 with the full observed series.

347 To ensure fair comparison, all optimal calculations are performed using the same multi-step forecast  
348 horizons as those used by the deep learning models in our experiments. This approach allows us to  
349 quantify precisely how close each forecasting model comes to the theoretical performance ceiling  
350 for each type of time series pattern.

## 351 E Models Forecasting Visualization

352 In this subsection, we provide visual comparisons of model forecasting performance across a diverse  
353 range of time series patterns. Figure 20 to Figure 23 show the prediction results of various forecast-  
354 ing models on representative datasets. We visualize model behavior on fundamental trend patterns  
355 (linear and exponential trends in Figure 20), periodic signals (Square-Wave and Triple-Sin patterns  
356 in Figure 21), and the impact of varying noise levels on both periodic and trend patterns (Double-Sin  
357 and Linear-Trend in Figure 24 and Figure 25). Additionally, we showcase forecasting capabilities  
358 on time series with temporal dependency characteristics (ARMA(1,1) process and Random Walk in  
359 Figure 22) and on more complex simulated real-world patterns (Network Traffic and Temperature  
360 Sensor data in Figure 23). These visualizations complement our quantitative analyses by providing  
361 intuitive representations of how different architectures handle various temporal patterns.

## 362 F Limitations

363 While our synthetic benchmark framework offers valuable insights into model capabilities, sev-  
364 eral limitations should be acknowledged. First, our generated time series, despite their complexity,

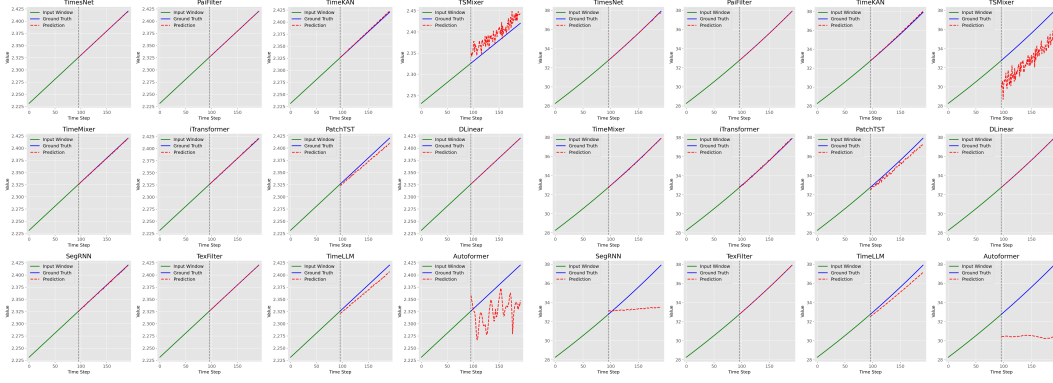


Figure 20: Comparison of model predictions for different trend patterns. The left panels show predictions for linear trend, while the right panels show predictions for exponential trend. Each subplot represents a different model’s prediction (red dashed line) compared to ground truth (blue solid line) and input window (green solid line).

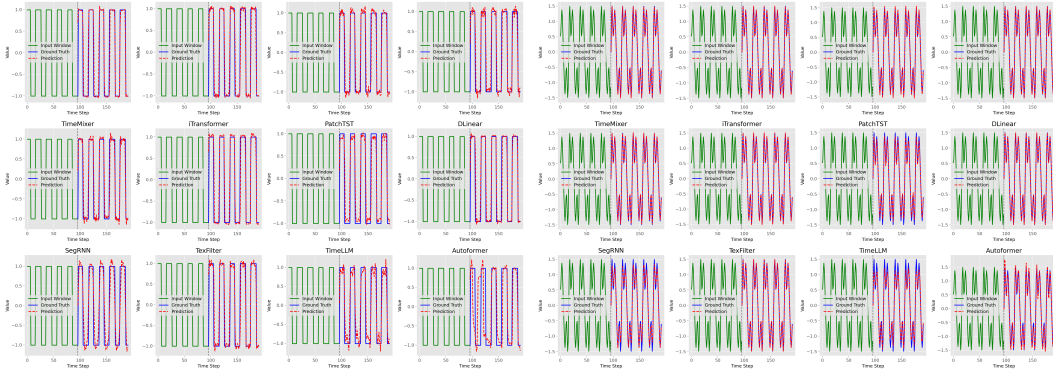


Figure 21: Comparison of model predictions for different periodic patterns. The left panels show predictions for Square-Wave patterns, while the right panels show predictions for Triple-Sin patterns. Each subplot represents a different model’s prediction (red dashed line) compared to ground truth (blue solid line) and input window (green solid line).

365 may not fully capture the intricacies of real-world data where multiple factors interact through so-  
 366 phisticated, non-linear mechanisms rather than simple superposition of components. Second, our  
 367 current framework does not address time-varying patterns where the underlying dynamics evolve  
 368 over time—a common characteristic in many real-world systems that we plan to explore in future  
 369 work. Third, due to computational constraints, we conducted all experiments with a fixed input  
 370 window size of 96 time steps, which, although standard in the field, limits our understanding of how  
 371 different temporal contexts might affect model performance.

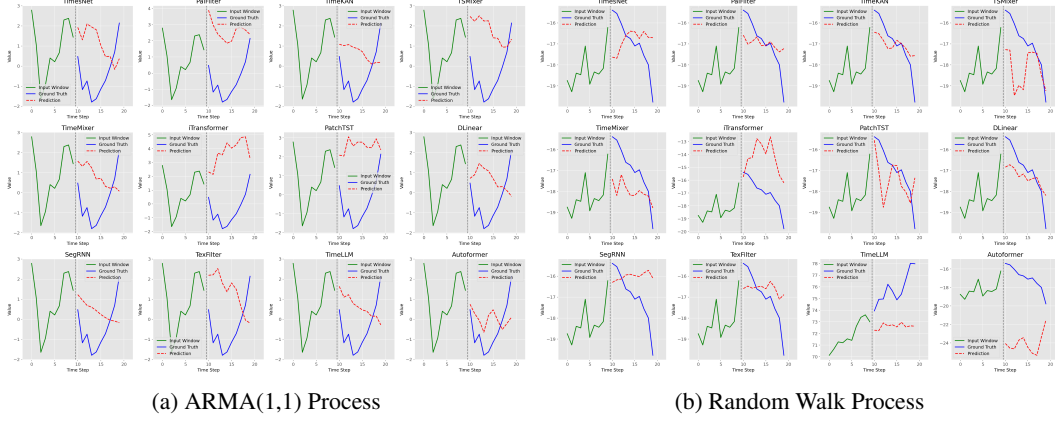


Figure 22: Model forecasting comparison on time series with temporal dependency characteristics. Left panels show predictions for ARMA(1,1) process with autocorrelation; right panels show predictions for Random Walk pattern commonly observed in financial data. Each subplot displays a model’s prediction (red dashed line) against the ground truth (blue solid line) with the input window (green solid line).

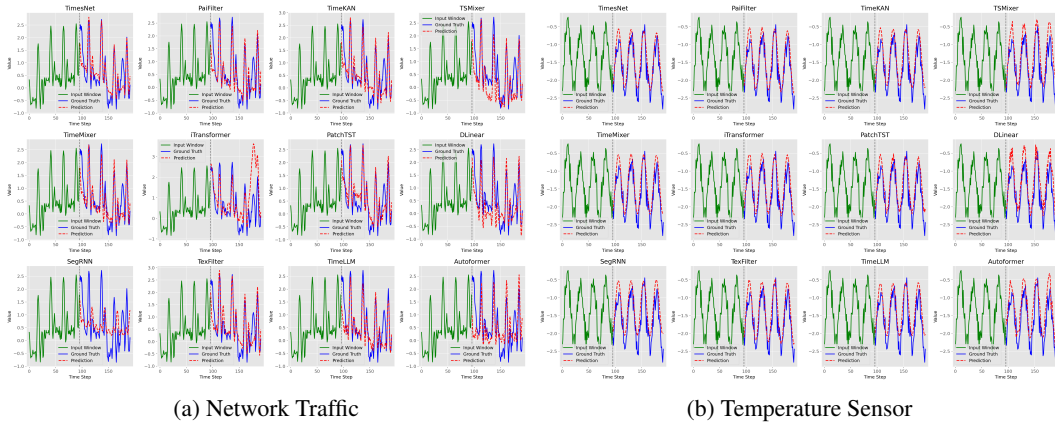


Figure 23: Model forecasting performance on simulated real-world time series data. Left panels show predictions for Network Traffic; right panels show predictions for Temperature Sensor. Each subplot displays a model’s prediction (red dashed line) against the ground truth (blue solid line) with the input window (green solid line).

Model Comparison Across Noise Levels (Double-Sin)

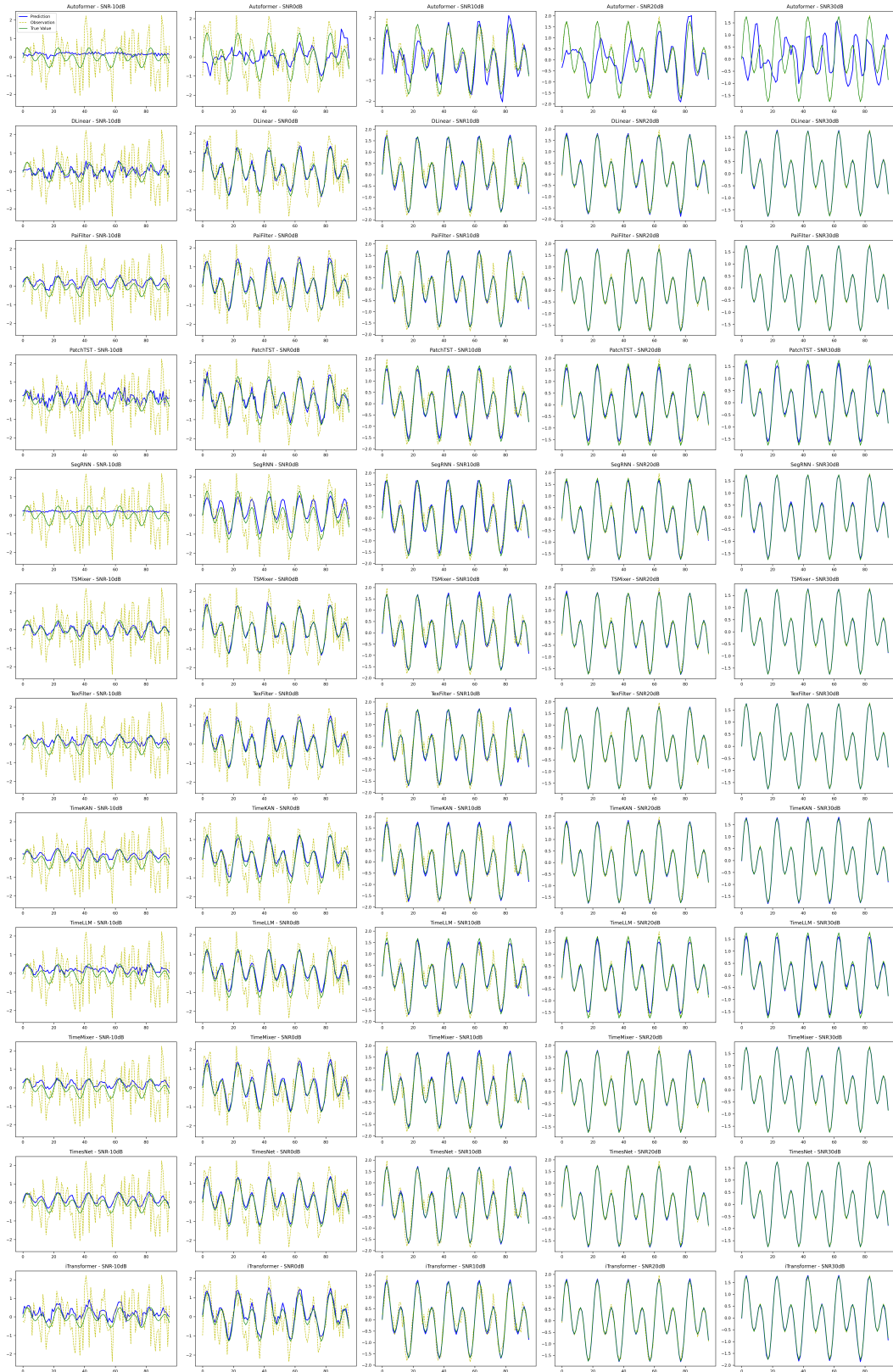


Figure 24: Comparison of model predictions under different noise levels for all models (Double-Sin). Blue lines represent model predictions, green lines show the true underlying signal, and yellow dashed lines represent the noisy observations.

Model Comparison Across Noise Levels (Linear-Trend)

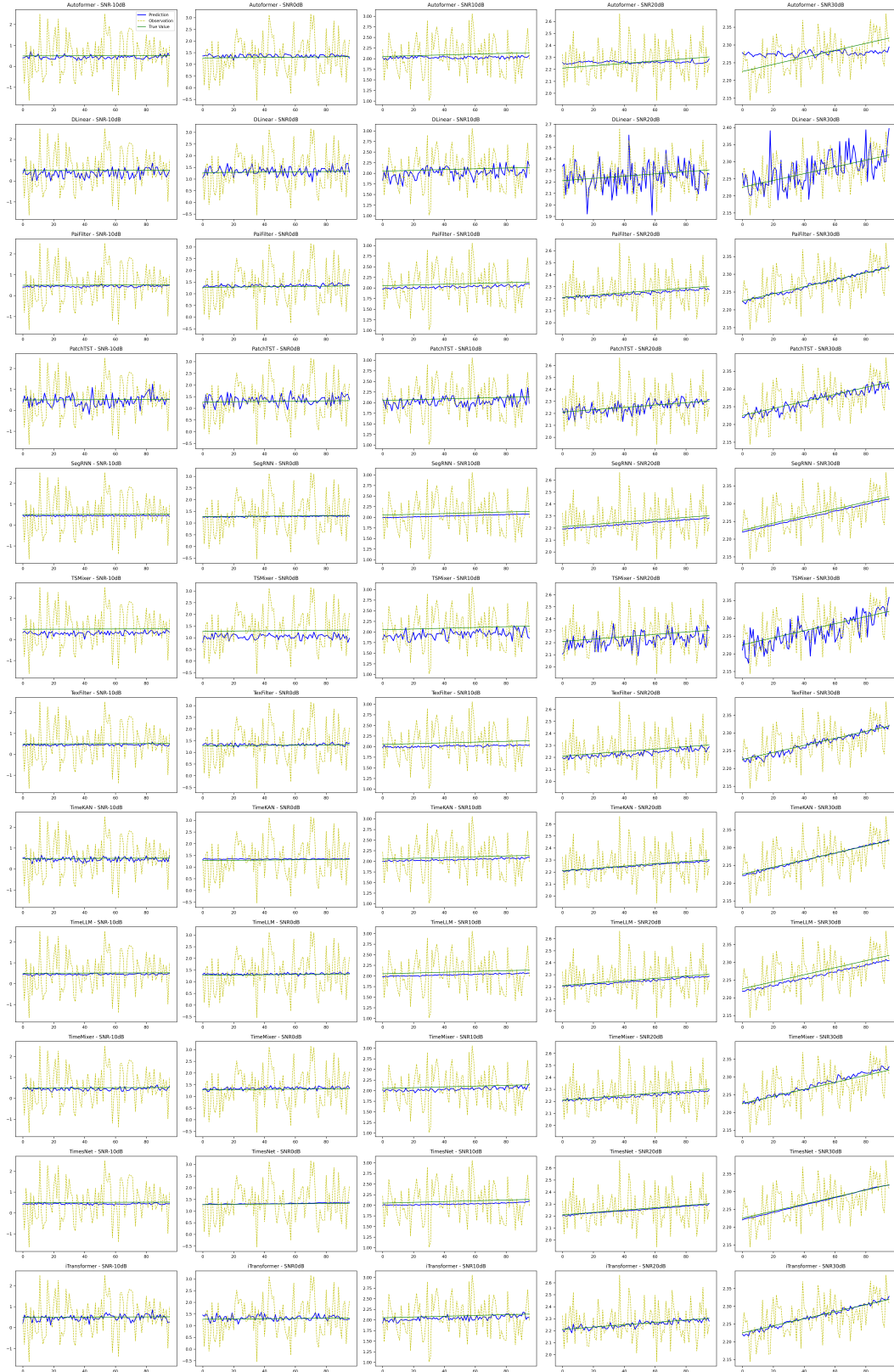


Figure 25: Comparison of model predictions under different noise levels for all models (Linear-Trend). Blue lines represent model predictions, green lines show the true underlying signal, and yellow dashed lines represent the noisy observations.



## References

- [1] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015. *arXiv:2311.10122*.
- [2] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.
- [3] Hiro Y Toda and Peter CB Phillips. Vector autoregression and causality: a theoretical overview and simulation study. *Econometric reviews*, 13(2):259–285, 1994.
- [4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, page 785–794. ACM, August 2016.
- [5] Lingyu Zhang, Wenjie Bian, Wenyi Qu, Liheng Tuo, and Yunhai Wang. Time series forecast of sales volume based on xgboost. In *Journal of Physics: Conference Series*, volume 1873, page 012067. IOP Publishing, 2021.
- [6] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
- [7] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.
- [8] Boris N. Oreshkin, Dmitri Carпов, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting, 2020.
- [9] A. Zeng, M. Chen, L. Zhang, et al. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023. *arXiv:2305.17328*.
- [10] 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, 2022.
- [11] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- [12] Z. C. Lipton. A critical review of recurrent neural networks for sequence learning. *arXiv Preprint*, CoRR(abs/1506.00019), 2015.
- [13] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International journal of forecasting*, 36(1):75–85, 2020.
- [14] Xueli Zhang, Cankun Zhong, Jianjun Zhang, Ting Wang, and Wing WY Ng. Robust recurrent neural networks for time series forecasting. *Neurocomputing*, 526:143–157, 2023.
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [16] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.
- [17] J. Chung, C. Gulcehre, K. H. Cho, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [18] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [19] Shuai Gao, Yuefei Huang, Shuo Zhang, Jingcheng Han, Guangqian Wang, Meixin Zhang, and Qingsheng Lin. Short-term runoff prediction with gru and lstm networks without requiring time step optimization during sample generation. *Journal of Hydrology*, 589:125188, 2020.

- [20] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.
- [21] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting, 2023.
- [22] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.
- [23] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.
- [24] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.
- [25] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis, 2023.
- [26] H. Zhou, S. Zhang, J. Peng, et al. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021. Systems, 2017.*arXiv:2306.02858*.
- [27] H. Wu, J. Xu, J. Wang, et al. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.
- [28] Y. Nie, N. H. Nguyen, P. Sinthong, et al. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.
- [29] 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.
- [30] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *# PLACEHOLDER\_PARENT\_METADATA\_VALUE#*, 2022.
- [31] 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.
- [32] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting, 2023.
- [33] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting, 2024.
- [34] Kun Yi, Jingru Fei, Qi Zhang, Hui He, Shufeng Hao, Defu Lian, and Wei Fan. Filternet: Harnessing frequency filters for time series forecasting, 2024.
- [35] 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.
- [36] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

- [37] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.
- [38] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. *arXiv preprint arXiv:2402.02368*, 2024.
- [39] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024.
- [40] Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: Building a unified time series model. *arXiv e-prints*, pages arXiv-2403, 2024.
- [41] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts, 2025.
- [42] Spyros Makridakis and Michele Hibon. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.
- [43] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of forecasting*, 34(4):802–808, 2018.
- [44] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.
- [45] Yubo Liang, Zezhi Shao, Fei Wang, Zhao Zhang, Tao Sun, and Yongjun Xu. Basics: An open source fair multivariate time series prediction benchmark. In *Bench*, pages 87–101, 2022.
- [46] Zezhi Shao, Fei Wang, Yongjun Xu, Wei Wei, Chengqing Yu, Zhao Zhang, Di Yao, Tao Sun, Guangyin Jin, Xin Cao, Gao Cong, Christian S. Jensen, and Xueqi Cheng. Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis, 2024.
- [47] Jiawen Zhang, Xumeng Wen, Zhenwei Zhang, Shun Zheng, Jia Li, and Jiang Bian. ProBTs: Benchmarking point and distributional forecasting across diverse prediction horizons. In *NeurIPS Datasets and Benchmarks Track*, 2024.
- [48] Yuxuan Wang, Haixu Wu, Jiayang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. 2024.
- [49] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S. Jensen, Zhenli Sheng, and Bin Yang. Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. *Proc. VLDB Endow.*, 17(9):2363–2377, 2024.
- [50] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models, 2024.
- [51] Songtao Huang, Zhen Zhao, Can Li, and Lei Bai. Timekan: Kan-based frequency decomposition learning architecture for long-term time series forecasting, 2025.