Supplementary Material: *Koopa: Learning Non-stationary Time Series Dynamics with Koopman Predictors*

Anonymous Author(s)
Affiliation
Address
`email`

1 Scaling Up Forecast Horizon

In this section, we introduce the capability of Koopa to scale up forecast horizon. In detail, we train a Koopa model with forecast length H_{tr} and attempt to apply it on a larger length H_{te}. The basic approach conducts rolling forecast by taking the model prediction as the input of the next iteration until the desired forecast horizon is all filled. Instead, we further assume that after the model gives a prediction, the model can utilize the incoming ground truth for *model adaptation* and continue rolling forecast for the next iteration. It is notable that we do not retrain parameters during model adaptation, since it will lead to overfitting on the incoming ground truth and Catastrophic Forgetting [3, 5, 10].

Koopa can naturally cope with the scenario by learning Koopman embedding and operator K_{inv} in Time-invariant KPs while calculating localized operator K_{var} to describe the dynamics in the temporal neighborhood. Therefore, we freeze the parameters of Koopa but only use the incoming ground truth for operator adaptation of K_{var} in Time-variant KPs.

1.1 Implementation of Operator Adaptation

At the beginning of rolling forecast, the Encoder in Time-variant KP outputs D-dimensional Koopman embedding for each observed series segment as $[z_1, z_2, \dots, z_F]$, where $F = \frac{H_{\text{tr}}}{S}$ is the segment number with S as the segment length. The operator K_{var} in Time-variant KP is calculated as follows:

$$Z_{\text{back}} = [z_1, z_2, \dots, z_{F-1}],\ Z_{\text{fore}} = [z_2, z_3, \dots, z_F],\ K_{\text{var}} = Z_{\text{fore}} Z_{\text{back}}^{\dagger}, \tag{1}$$

where $Z_{\text{back}}, Z_{\text{fore}} \in \mathbb{R}^{D \times (F-1)}, K_{\text{var}} \in \mathbb{R}^{D \times D}$. With the calculated operator, we obtain the next predicted Koopman embedding by one-step forwarding:

$$\hat{z}_{F+1} = K_{\text{var}} z_F. \tag{2}$$

After decoding the embedding $\hat{z}_{F+1}$ to the series prediction, we can utilize the true value of incoming Koopman embedding z_{F+1} obtained by Koopa with frozen parameters. Instead of using K_{var} to obtain the next embedding $\hat{z}_{F+2}$, we use incremental embedding collections $Z_{\text{back}+}, Z_{\text{fore}+} \in \mathbb{R}^{D \times F}$ to obtain a more accurate operator $K_{\text{var}+} \in \mathbb{R}^{D \times D}$ to describe the local dynamics:

$$Z_{\text{back}+} = [Z_{\text{back}}, z_F],\ Z_{\text{fore}+} = [Z_{\text{back}}, z_{F+1}],\ K_{\text{var}+} = Z_{\text{fore}+} Z_{\text{back}+}^{\dagger}. \tag{3}$$

The procedure repeats for L times ($L \propto H_{\text{te}}$) until the forecast horizon is all filled, we formulate it as Algorithm 1. And experimental results (Koopa OA) in the Section 5.3 of the main text have demonstrated the promotion of forecasting performance due to more precisely fitted dynamics.

1.2 Computational Acceleration

The naïve implementation shown in Algorithm 1 repeatedly conducts Equation 3 on the incremental embedding collection to obtain new operators, which has a complexity of $\mathcal{O}(LD^3)$. We propose an equivalent algorithm with improved complexity of $\mathcal{O}((L+D)D^2)$ as shown in Algorithm 2.

Theorem. *Algorithm 2 gives the same K_{var} as Algorithm 1 in each iteration with $\mathcal{O}(D^2)$ complexity.*

Proof. We start with the first iteration analysis. By the definition of Moore–Penrose inverse, we have $Z_{\text{back}}^{\dagger} Z_{\text{back}} = I_{F-1}$, where I_{F-1} is an identity matrix with the dimension of $F-1$. When the model receives the incoming embedding z_{F+1}, incremental embedding $m = z_F, n = z_{F+1}$ will be appended to Z_{back} and Z_{fore} respectively. Instead of calculating new $K_{\text{var+}}$ from incremental collections, we utilize calculated K_{var} to find the iteration rule on $K_{\text{var+}}$. Concretely, we suppose

$$Z_{\text{back}+}^{\dagger} = \begin{bmatrix} Z_{\text{back}}^{\dagger} - \Delta \\ b^{\top} \end{bmatrix} \in \mathbb{R}^{F \times D}, \tag{4}$$

where $\Delta \in \mathbb{R}^{(F-1)\times D}, b \in \mathbb{R}^D$ are variables to be identified. By the definition of Moore–Penrose inverse, we have $Z_{\text{back}+}^{\dagger} Z_{\text{back}+} = I_F$. By unfolding it, we have the following equations:

$$\Delta Z_{\text{back}} = \mathbf{0},\ b^{\top} Z_{\text{back}} = \vec{0},\ b^{\top} m = 1,\ Z_{\text{back}}^{\dagger} m - \Delta m = \vec{0}. \tag{5}$$

We suppose $\Delta = \delta b^{\top}$, where $\delta \in \mathbb{R}^{F-1}$, such that when $b^{\top} Z_{\text{back}} = \vec{0}$, then $\Delta Z_{\text{back}} = \mathbf{0}$. Then we have $Z_{\text{back}}^{\dagger} m - \delta b^{\top} m = Z_{\text{back}}^{\dagger} m - \delta = \vec{0}$, thus $\Delta = Z_{\text{back}}^{\dagger} m b^{\top}$. Given equations that $b^{\top} Z_{\text{back}} = \vec{0}$ and $b^{\top} m = 1$, we have the analytical solution of b:

$$b = r/||r||^2, \text{ where } r = m - Z_{\text{back}} Z_{\text{back}}^{\dagger} m. \tag{6}$$

Therefore, we find the equation between the incremental version $K_{\text{var+}}$ and calculated K_{var}:

$$Z_{\text{back}+}^{\dagger} = \begin{bmatrix} Z_{\text{back}}^{\dagger}(I_D - m b^{\top}) \\ b^{\top} \end{bmatrix},\ K_{\text{var+}} = Z_{\text{fore+}} Z_{\text{back}+}^{\dagger} = K_{\text{var}} + (n - K_{\text{var}} m) b^{\top}, \tag{7}$$

where m, n are the incremental embedding of $Z_{\text{back}}, Z_{\text{fore}}$ and b can be calculated by Equation 6. We also derive the iteration rule on $X = Z_{\text{back}} Z_{\text{back}}^{\dagger}$ to obtain b, which is formulated as follows:

$$X_{+} = Z_{\text{back+}} Z_{\text{back+}}^{\dagger} = X + (m - Xm) b^{\top} = X + r b^{\top}. \tag{8}$$

By adopting Equation 7– 8 and permuting the matrix multiplication order, we reduce the complexity of each iteration to $\mathcal{O}(D^2)$. Therefore, Algorithm 2 has a overall complexity of $\mathcal{O}((L+D)D^2)$. Since $L \propto H_{\text{te}}$, Algorithm 1–2 have $\mathcal{O}(H_{\text{te}} D^3)$ and $\mathcal{O}((H_{\text{te}} + D)D^2)$ complexity respectively.

Algorithm 1 Koopa Operator Adaptation.

Require: Observed embedding $Z = [z_1, \dots, z_F]$ and successively incoming ground truth embedding $[z_{F+1}, \dots, z_{F+L}]$ with each embedding $z_i \in \mathbb{R}^D$.

$Z_{\text{back}} = [z_1, \dots, z_{F-1}], Z_{\text{fore}} = [z_2, \dots, z_F]$ ▷ $Z_{\text{back}}, Z_{\text{fore}} \in \mathbb{R}^{D \times (F-1)}$

$K_{\text{var}} = Z_{\text{fore}} Z_{\text{back}}^{\dagger}$ ▷ $K_{\text{var}} \in \mathbb{R}^{D \times D}$

$\hat{z}_{F+1} = K_{\text{var}} n$ ▷ $\hat{z}_{F+1} \in \mathbb{R}^D$

for l **in** $\{1, \dots, L\}$**:** ▷ z_{F+l} comes successively

  $m = z_{F+l-1}, n = z_{F+l}$ ▷ $m, n \in \mathbb{R}^D$

  $Z_{\text{back}} \leftarrow [Z_{\text{back}}, m], Z_{\text{fore}} \leftarrow [Z_{\text{fore}}, n]$ ▷ $Z_{\text{back}}, Z_{\text{fore}} \in \mathbb{R}^{D \times (F+l-1)}$

  $K_{\text{var}} = Z_{\text{fore}} Z_{\text{back}}^{\dagger}$ ▷ $K_{\text{var}} \in \mathbb{R}^{D \times D}$

  $\hat{z}_{F+l+1} = K_{\text{var}} n$ ▷ $\hat{z}_{F+l+1} \in \mathbb{R}^D$

End for

Return $[\hat{z}_{F+1}, \dots, \hat{z}_{F+L+1}]$ ▷ Return predicted embedding

---

Algorithm 2 Accelerated Koopa Operator Adaptation.

Require: Observed embedding $Z = [z_1, \dots, z_F]$ and successively incoming ground truth embedding $[z_{F+1}, \dots, z_{F+L}]$ with each embedding $z_i \in \mathbb{R}^D$.

```
Z_back = [z_1, ..., z_{F-1}], Z_fore = [z_2, ..., z_F]        ▷ Z_back, Z_fore ∈ R^{D×(F−1)}
K_var = Z_fore Z_back^†, X = Z_back Z_back^†                  ▷ K_var, X ∈ R^{D×D}
ẑ_{F+1} = K_var n                                             ▷ ẑ_{F+1} ∈ R^D
for l in {1, ..., L}:                                         ▷ z_{F+l} comes successively
    m = z_{F+l−1}, n = z_{F+l}                                ▷ m, n ∈ R^D
    r = m − Xm                                                ▷ r ∈ R^D
    b = r/||r||^2                                             ▷ b ∈ R^D
    K_var ← K_var + (n − K_var m) b^⊤                         ▷ K_var ∈ R^{D×D}
    X ← X + r b^⊤                                             ▷ X ∈ R^{D×D}
    ẑ_{F+l+1} = K_var n                                       ▷ ẑ_{F+l+1} ∈ R^D
End for
Return [ẑ_{F+1}, ..., ẑ_{F+L+1}]^⊤                            ▷ Return predicted embedding
```

2 Implementation Details

Koopa is trained with L2 loss and optimized by ADAM [4] with an initial learning rate of 0.001 and batch size set to 32. The training process is early stopped within 10 epochs. We repeat each experiment three times with different random seeds to obtain average test MSE/MAE and detailed results with standard deviations are listed in Table 1. All experiments are implemented in PyTorch [9] and conducted on NVIDIA TITAN RTX 24GB GPUs.

All the baselines that we reproduced are implemented based on the benchmark of TimesNet [14] Repository, which is fairly built on the configurations provided by each model's original paper or official code. Since several baselines adopt Series Stationarization from Non-stationary Transformers [6] while others do not, we equip all models with the method for a fair comparison.

Table 1: Detailed performance of Koopa. We report the MSE/MAE and standard deviation of different forecast horizons $\{H_1, H_2, H_3, H_4\} = \{24, 36, 48, 60\}$ for ILI and $\{48, 96, 144, 192\}$ for others.

Dataset	ECL		ETTh2		Exchange	
Horizon	MSE	MAE	MSE	MAE	MSE	MAE
H_1	0.130±0.003	0.234±0.003	0.226±0.003	0.300±0.003	0.042±0.002	0.143±0.003
H_2	0.136±0.004	0.236±0.005	0.297±0.004	0.349±0.004	0.083±0.004	0.207±0.004
H_3	0.149±0.003	0.247±0.003	0.333±0.004	0.381±0.003	0.130±0.005	0.261±0.003
H_4	0.156±0.004	0.254±0.003	0.356±0.005	0.393±0.004	0.184±0.009	0.309±0.005
Dataset	ILI		Traffic		Weather	
Horizon	MSE	MAE	MSE	MAE	MSE	MAE
H_1	1.621±0.008	0.800±0.006	0.415±0.003	0.274±0.005	0.126±0.005	0.168±0.004
H_2	1.803±0.040	0.855±0.020	0.401±0.005	0.275±0.004	0.154±0.006	0.205±0.003
H_3	1.768±0.015	0.903±0.008	0.397±0.004	0.276±0.003	0.172±0.005	0.225±0.005
H_4	1.743±0.040	0.891±0.009	0.403±0.007	0.284±0.009	0.193±0.003	0.241±0.004

3 Hyperparameter Sensitivity

We verify the robustness of Koopa with respect to hyperparameters as follows: the dimension of Koopman embedding D, the hidden layer number l and the hidden dimension d used in Encoder and

Decoder. Considering the efficiency of hyperparameters search, we fix the segment length $S = T/2$ and the number of Koopa blocks $B = 3$. As the detailed hyperparameter sensitivity analysis is shown in Figure 1, we find the proposed model is insensitive to the choices of above hyperparameters, which can be beneficial for practitioners to reduce hyperparameters search burden in real-world applications.

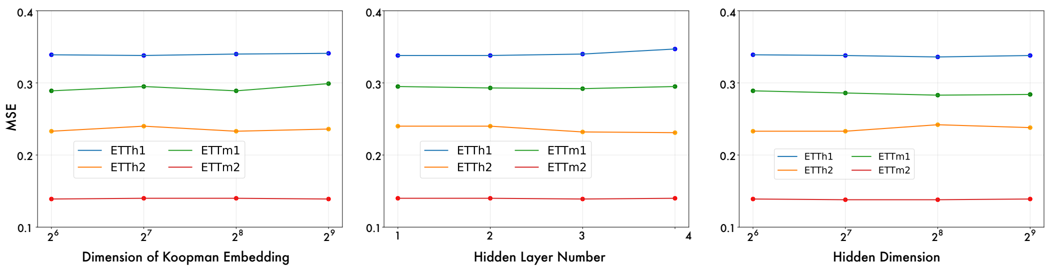

Figure 1: Hyperparameter sensitivity with respect to the dimension of Koopman embedding, hidden layer number, and hidden dimension of Encoder and Decoder in Koopa. The results are recorded with forecast window length $H = 48$ in ETT datasets.

4 Supplementary Experimental Results

4.1 Full Forecasting Results

Due to the limited pages, we list additional multivariate benchmarks on ETT datasets [17] in Table 2, which includes the hourly recorded ETTh2 and 15-minutely recorded ETTm1/ETTm2, and the full univariate results of M4 [11] in Table 3, which contains the yearly, quarterly and monthly collected univariate marketing data. Notably, Koopa still achieves competitive performance compared with state-of-the-art deep forecasting models and specialized univariate models.

Table 2: Forecasting results with different forecast window lengths $H \in \{48, 96, 144, 192\}$ on ETT dataset. We set the lookback window length $T = 2H$.

Models		KooPA		PatchTST [7]		TimesNet [14]		DLinear [16]		MICN [12]		KNF [13]		FiLM [18]		Autoformer [15]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	48	**0.283**	**0.333**	0.286	0.336	0.308	0.354	0.322	0.355	0.294	0.353	1.026	0.792	0.324	0.353	0.592	0.419
	96	**0.294**	**0.345**	0.299	0.346	0.329	0.370	0.309	0.346	0.306	0.364	0.957	0.782	0.311	0.346	0.493	0.469
	144	**0.322**	0.366	0.325	0.363	0.358	0.387	0.327	**0.359**	0.342	0.390	0.921	0.760	0.328	0.358	0.735	0.569
	192	**0.337**	0.378	0.343	0.375	0.462	0.441	**0.337**	**0.365**	0.386	0.415	0.896	0.731	0.339	0.366	0.592	0.506
ETTm2	48	0.134	**0.226**	0.135	0.231	0.142	0.234	0.144	0.240	**0.131**	0.238	0.621	0.623	0.146	0.243	0.191	0.280
	96	**0.171**	**0.254**	**0.171**	0.255	0.187	0.269	0.172	0.256	0.197	0.295	1.535	1.012	0.174	0.257	0.241	0.311
	144	0.206	0.280	0.205	0.282	0.216	0.291	**0.200**	**0.276**	0.210	0.297	1.337	0.876	0.204	0.279	0.300	0.352
	192	0.226	0.298	0.221	0.294	0.243	0.313	**0.219**	**0.290**	0.248	0.328	1.355	0.908	0.224	0.293	0.324	0.370
ETTh1	48	**0.336**	0.377	0.337	0.375	0.365	0.399	0.343	**0.371**	0.375	0.406	0.876	0.709	0.407	0.427	0.442	0.438
	96	**0.371**	0.405	0.372	**0.393**	0.411	0.430	0.379	**0.393**	0.406	0.429	0.975	0.744	0.429	0.431	0.634	0.523
	144	0.405	0.418	0.394	0.412	0.442	0.447	**0.393**	**0.403**	0.437	0.448	0.801	0.662	0.451	0.448	0.522	0.491
	192	0.416	0.429	0.416	0.439	0.469	0.470	**0.407**	**0.416**	0.518	0.496	0.941	0.744	0.460	0.459	0.525	0.501

4.2 Full Ablation Results

We elaborately conduct model ablations in Table 4 to verify the effect of our proposed modules: Time-invariant KP, Time-variant KP, Fourier Filter and the choices to tackle dynamics.

Dynamics underlying time series As shown in Table 4, Time-variant and Time-invariant KPs perform as complementary modules for exploring the dynamics underlying time series, discarding any one of them (Only K_{inv} and Only K_{var}) will lead to the inferior forecasting performance. It is also a surprising finding that only utilizing Time-invariant KP surpasses only utilizing Time-variant KP in more cases (ECL, ETTh2, Exchange, Traffic), indicating the time-variant dynamics plays a

Table 3: Full univariate forecasting results for M4 dataset. We follow the same data processing and forecasting length settings used in TimesNet [14] benchmark. Additional forecasting models N-HiTS [1] and N-BEATS [8] are also included.

Models		KooPA	N-HiTS	N-BEATS	PatchTST	TimesNet	DLinear	MICN	KNF	FiLM	Autoformer
Year	sMAPE	**13.352**	13.371	13.466	13.517	13.394	13.866	14.532	13.986	14.012	14.786
	MASE	**2.997**	3.025	3.059	3.031	3.004	3.006	3.359	3.029	3.071	3.349
	OWA	**0.786**	0.790	0.797	0.795	0.787	0.802	0.867	0.804	0.815	0.874
Quarter	sMAPE	10.159	10.454	**10.074**	10.847	10.101	10.689	11.395	10.343	10.758	12.125
	MASE	1.189	1.219	**1.163**	1.315	1.183	1.294	1.379	1.202	1.306	1.483
	OWA	0.895	0.919	**0.881**	0.972	0.890	0.957	1.020	0.965	0.905	1.091
Month	sMAPE	**12.730**	12.794	12.801	14.584	12.866	13.372	13.829	12.894	13.377	15.530
	MASE	**0.953**	0.960	0.955	1.169	0.964	1.014	1.082	1.023	1.021	1.277
	OWA	0.901	0.895	**0.893**	1.055	0.894	0.940	0.988	0.985	0.944	1.139
Others	sMAPE	4.861	**4.696**	5.008	6.184	4.982	4.894	6.151	4.753	5.259	5.841
	MASE	**3.124**	3.130	3.443	4.818	3.323	3.358	4.263	3.138	3.608	4.308
	OWA	1.004	**0.988**	1.070	1.140	1.048	1.044	1.319	1.019	1.122	1.294
Weighted Average	sMAPE	**11.863**	11.960	11.910	13.022	11.930	12.418	13.023	12.126	12.489	14.057
	MASE	**1.595**	1.606	1.613	1.814	1.597	1.656	1.836	1.641	1.690	1.954
	OWA	**0.858**	0.861	0.862	0.954	0.867	0.891	0.960	0.874	0.902	1.029

Table 4: Model ablation with detailed forecasting performance. We report forecasting results with different prediction lengths $\{24, 36, 48, 60\}$ for ILI and $H \in \{48, 96, 144, 192\}$ for others. For columns: Only K_{inv} uses one-block Time-invariant KP; Only K_{var} stacks Time-variant KPs only; *Truncated Filter* replaces Fourier Filter with High-Low Frequency Pass Filter; *Branch Switch* changes the order of KPs to deal with disentangled components.

Models		**KooPA**		Only K_{inv}		Only K_{var}		Truncated Filter		Branch Switch	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	48	**0.130**	**0.234**	0.150	0.243	1.041	0.777	0.149	0.245	0.137	0.234
	96	**0.136**	**0.236**	0.137	0.242	4.643	1.669	0.172	0.280	2.240	0.724
	144	**0.149**	0.247	0.150	0.252	0.238	0.327	**0.149**	**0.246**	0.226	0.331
	192	0.156	0.254	0.158	0.260	0.267	0.355	**0.152**	**0.248**	0.181	0.284
ETTh2	48	**0.226**	**0.300**	0.235	0.304	0.271	0.334	0.340	0.310	0.245	0.317
	96	**0.297**	**0.349**	0.311	0.353	0.382	0.405	0.301	0.352	0.343	0.384
	144	**0.333**	0.381	0.337	**0.379**	0.427	0.444	0.338	0.386	0.403	0.418
	192	**0.356**	**0.393**	0.363	0.397	0.402	0.437	0.363	0.400	0.384	0.420
Exchange	48	**0.042**	**0.143**	0.046	0.150	0.065	0.184	0.048	0.150	0.055	0.165
	96	**0.083**	**0.207**	**0.083**	0.210	0.147	0.274	0.087	0.210	0.151	0.277
	144	**0.130**	**0.261**	0.149	0.281	0.222	0.351	0.150	0.278	0.254	0.369
	192	**0.184**	**0.309**	0.200	0.322	0.385	0.456	0.229	0.345	0.463	0.490
ILI	24	**1.621**	**0.800**	2.165	0.882	1.972	0.919	2.140	0.874	2.092	0.894
	36	**1.803**	0.855	1.815	0.882	2.675	1.091	1.692	**0.844**	2.116	0.950
	48	**1.768**	0.903	2.107	0.981	2.446	1.045	1.762	**0.895**	2.394	1.084
	60	**1.743**	**0.891**	2.496	1.108	2.387	0.970	2.357	1.018	1.917	0.926
Traffic	48	**0.415**	**0.274**	0.445	0.295	0.915	0.536	0.668	0.363	0.468	0.300
	96	**0.401**	**0.275**	0.403	0.277	0.833	0.465	0.441	0.323	0.429	0.298
	144	**0.397**	**0.276**	0.400	0.278	0.816	0.452	0.436	0.321	0.438	0.307
	192	**0.403**	**0.284**	1.371	0.788	1.224	0.723	0.597	0.331	0.469	0.312
Weather	48	0.126	0.168	0.142	0.181	0.140	0.190	**0.125**	**0.166**	0.130	0.173
	96	0.154	0.205	0.164	0.209	0.169	0.224	**0.154**	**0.202**	0.163	0.210
	144	**0.172**	**0.225**	0.178	0.226	0.194	0.247	0.176	0.226	0.187	0.238
	192	**0.193**	**0.241**	0.195	0.245	0.217	0.268	0.195	0.244	0.212	0.261

dominant role in these time series datasets and it also emphasizes the significance to first establish the time-invariant dynamics and then assisted it with adapted local time-variant dynamics.

Appropriate disentanglement By replacing the proposed Fourier Filter with another disentangling method (Truncated Filter), we validate the effectiveness of our proposed modules to disentangle time-variant and time-invariant dynamics, since the amplitude statistics of frequency spectrums work as a global view to exhibit the time-agnostic information, and thus reveals perfectly the specific and common dynamics underlying each series window. Besides, Koopa fundamentally considers the properties of disentangled time series with the right Koopman Predictors, and switching the order of KPs (Branch Switch) will lead to degrading performance.

4.3 Model Efficiency

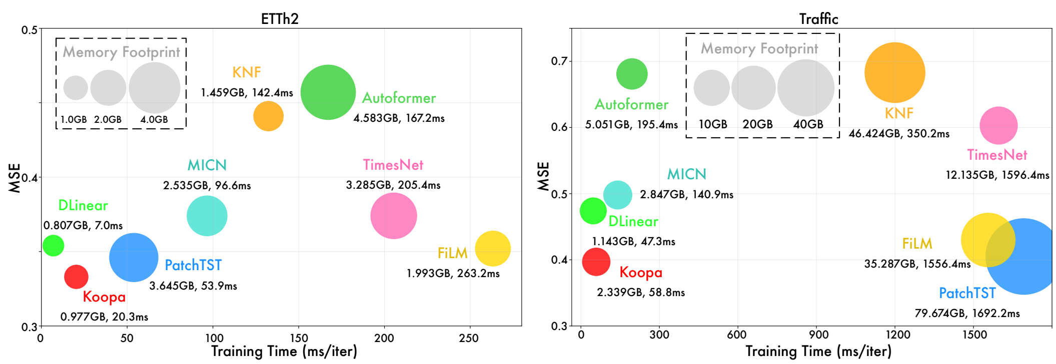

Figure 2: Model efficiency comparison with forecast length $H = 144$ for ETTh2 and Traffic.

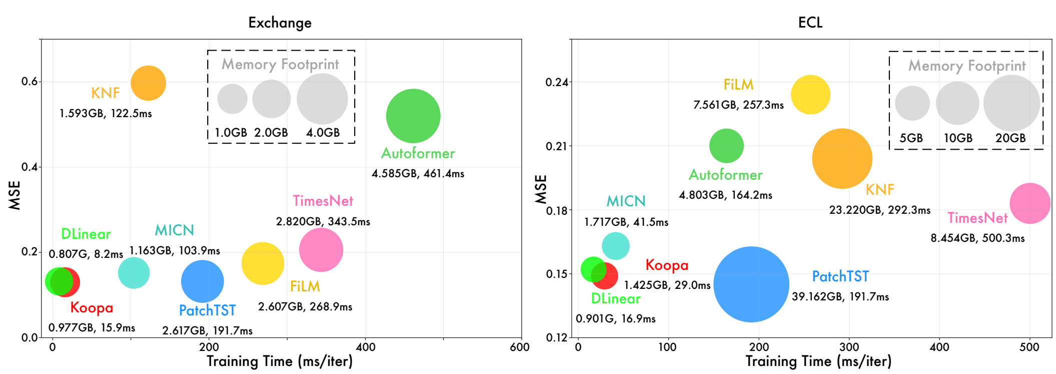

Figure 3: Model efficiency comparison with forecast length $H = 144$ for Exchange and ECL.

We comprehensively compare the forecasting performance, training speed, and memory footprint of our model with well-acknowledged deep forecasting models. The results are recorded with the official model configuration and the same batch size. We visualize the model efficiency under all six multivariate datasets in Figure 2– 4. In detail, compared with the previous state-of-the-art model PatchTST [2], Koopa consumes only 15.2% training time and 3.6% memory footprint respectively in ECL, 37.8% training time and 26.8% memory in ETTh2, 23.5% training time and 37.3% memory in Exchange, 50.9% training time and 47.8% memory in ILI, 3.5% training time and 2.9% memory in Traffic, and 5.4% training time and 25.4% memory in Weather, leading to the averaged **77.3%** and **76.0%** saving of training time and memory footprint in all six datasets. The remarkable efficiency can be attributed to Koopa with MLPs as the building blocks, and we find the budget saving becoming more significant on datasets with more series variables (ECL, Traffic).

Besides, as an efficient linear model, the performance of Koopa still surpasses other MLP-based models. Especially, Compared with DLinear [16], our model reduces 38.0% MSE (2.852→1.768) in

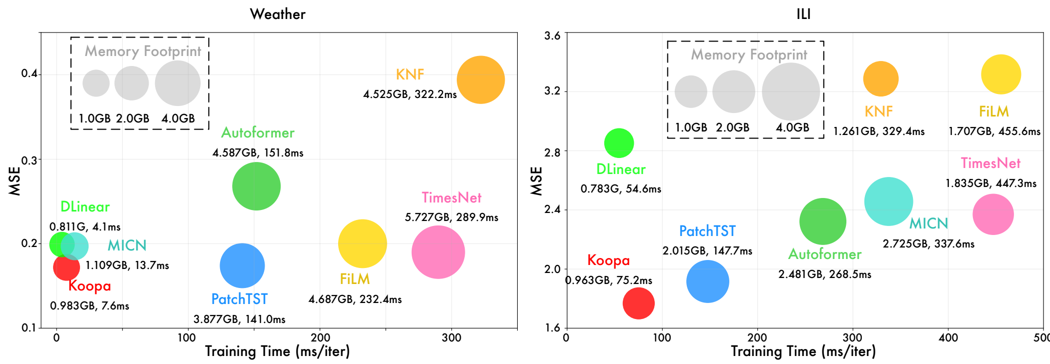

Figure 4: Model efficiency comparison with forecast length $H = 144$ for Weather and 48 for ILI.

ILI and 13.6% MSE (0.452→0.397) in Weather. And the average MSE reduction of Koopa compared with the previous state-of-the-art MLP-based model reaches **12.2%**.

Therefore, our proposed Koopa is efficiently built with MLP networks and shows great model capacity to exploit nonlinear dynamics and complicated temporal dependencies in real-world time series.

5 Broader Impact

5.1 Impact on Real-world Applications

This paper copes with real-world time series forecasting, which is characterized by intrinsic non-stationarity that poses fundamental challenges for deep forecasting models. Since previous studies hardly research the theoretical basis that can naturally address the time-variant property in non-stationary data, we propose a novel Koopman forecaster that fundamentally considers the implicit time-variant and time-invariant dynamics based on Koopman theory. Our model achieves the state-of-the-art performance on six real-world forecasting tasks, covering energy, economics, disease, traffic, and weather, and demonstrates remarkable model efficiency in training time and memory footprint. Therefore, the proposed model makes it promising to tackle real-world forecasting applications, which helps our society prevent risks in advance and make better decisions with limited computational budgets. Our paper mainly focuses on scientific research and has no obvious negative social impact.

5.2 Impact on Future Research

In this paper, we find modern Koopman theory natural to learn the dynamics underlying non-stationary time series. The proposed model explores complex non-stationary patterns with temporal localization inspired by Koopman approaches and implements respective deep network modules to disentangle and portray time-variant and time-invariant dynamics with the enlightenment of Wold's Theorem. The remarkable efficiency and insights from the theory can be instructive for future research.

6 Limitation

Our proposed model does not respectively considers dynamics in different variates, which leaves improvement for better multivariate forecasting with the consideration of various evolution patterns and series relationships. And Koopman spectral theory is still under leveraging in our work, which can discover Koopman modes to interpret the linear behavior underlying non-stationary data in a high-dimensional representation. Besides, Koopman theory for control considering factors outside the system can be promising for series forecasting with covariates, which leaves our future work.

References

[1] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886*, 2022.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[3] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[6] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Rethinking the stationarity in time series forecasting. *arXiv preprint arXiv:2205.14415*, 2022.

[7] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[8] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *ICLR*, 2019.

[9] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[10] Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

[11] Spyros Makridakis. M4 dataset, 2018.

[12] Huiqiang Wang, Jian Peng, Feihu Huang, Jince Wang, Junhui Chen, and Yifei Xiao. Micn: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

[13] Rui Wang, Yihe Dong, Sercan O Arik, and Rose Yu. Koopman neural forecaster for time series with temporal distribution shifts. *arXiv preprint arXiv:2210.03675*, 2022.

[14] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[15] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting. In *NeurIPS*, 2021.

[16] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

[17] 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. In *AAAI*, 2021.

[18] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, Rong Jin, et al. Film: Frequency improved legendre memory model for long-term time series forecasting. *Advances in Neural Information Processing Systems*, 35:12677–12690, 2022.