Enhancing Time Series Forecasting through Selective Representation Spaces: A Patch Perspective

Xingjian Wu, Xiangfei Qiu, Hanyin Cheng,
Zhengyu Li, Jilin Hu, Chenjuan Guo, Bin Yang*

華東師範大學

NEURAL INFORMATION PROCESSING SYSTEMS

Introduction

Conventional patching partitions a time series into adjacent patches, which causes a fixed representation space, thus resulting in insufficiently expressful representations. Because fixed representation spaces assume that all information useful for forecasting is evenly distributed in the contextual time series. As shown in Figure 1, the assumption is broken due to the phenomenons of changeable periods, shifting, and anomalies.

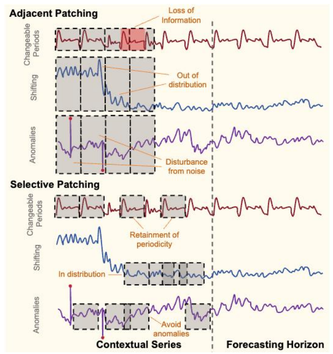

Figure 1: Adjacent vs. Selective Patching (both using 4 patches). Adjacent patching partitions time series into adjacent patches, lacking flexibility. Selective patching automatically select most relevant sub-series as patches. The upper part shows examples that conventional adjacent patching may include harmful information, thus hindering the forecasting performance. The lower part shows the selected patches that are more relevant for making the corresponding forecasting, demonstrating the flexibility that selective patches offer.

In this paper, we pioneer the exploration of constructing a selective representation space to flexibly include the most informative patches for forecasting. Specifically, we propose the Selective Representation Space (SRS) module, which utilizes the learnable Selective Patching and Dynamic Reassembly techniques to adaptively select and shuffle the patches from the contextual time series, aiming at fully exploiting the information of contextual time series to enhance the forecasting performance of patch-based models.

- We propose a modular SRS, which efficiently and adaptively constructs the selective representation space to fully exploit the information in the contextual time series, and is able to improve the performance of patch-based models in a simple plugin-and-play paradigm.
- Technically, we devise the Selective Patching and Dynamic Reassembly techniques, which are able to constitute a selective representation space at the patch perspective. And they are easily to be optimized through gradient-based strategies.
- Applying SRS with an MLP forms a simple-yet-effective method, called SRSNet. SRSNet achieves state-of-the-art performance across multiple real-world datasets, demonstrating the effectiveness of SRS module.

Paper:

Code:

Methodology

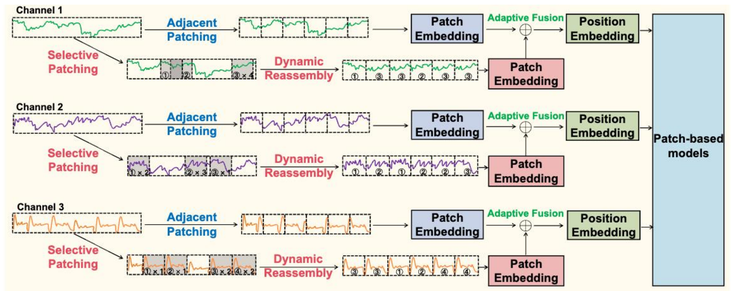

Figure 2: The overall pipeline of the SRS module. The multivariate time series is processed with Channel Independent strategy, the Selective Patching first adaptively chooses proper patches from all potential candidate patches. Then the Dynamic Reassembly dertermines the order of the selected patches. Both the Selective Patching and Dynamic Reassembly are gradient-based and learnable. Finally, the Adaptive Fusion integrates the embeddings from Conventional Patching and Dynamic Reassembly, adds the position embeddings to construct the final representations. The subsequent backbones can be used directly without changes, so that the SRS module is a modular plugin.

We propose gradient-based Selective Patching and Dynamic Reassembly modules, we take the Selective Patching as an example to introduce the core mechanism. As shown in Figure 3, we scan the time series with stride equals 1, where all potential patches are represented as

To support selection with replacement, we generate n scores for each patch, denoting the scores of n times of sampling:

$$\mathcal{P}' \in \mathbb{R}^{N\times K\times p}, K=(n-1)\cdot s+1.$$

$$\text{Scorer}^s := \mathbb{R}^{N\times K\times p} \to \mathbb{R}^{N\times K\times n},\ \mathcal{S}^s = \text{Scorer}^s(\mathcal{P}'), \mathcal{I}^s = \text{Argmax}(\mathcal{S}^s),$$

However, the Argmax operation interrupts the gradient propagation, and existing soft sorting methods keep gradient propagation but introduce noise and inaccuracies, making the sorting non-intuitive. To mitigate this, we devise a method to achieve differentiable sorting:

$$\mathcal{S}^s_{max} = \mathcal{S}^s[\mathcal{I}^s], \mathcal{S}^s_{inv} = \text{detach}(1/\mathcal{S}^s_{max}), \mathcal{P}^s_{max} = \mathcal{P}'[\mathcal{I}^s], E^s = \mathcal{S}^s_{max} \odot \mathcal{S}^s_{inv},$$

$$\tilde{\mathcal{P}}^s_{\text{max}} = \mathcal{P}^s_{max} \odot E^s,$$

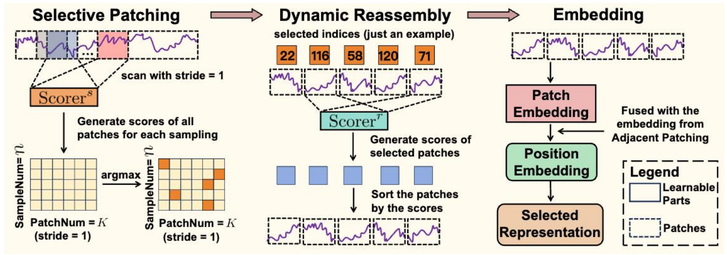

Figure 3: The detailed architecture of the SRS module. The Selective Patching allows sampling with replacement. It scans all the potential patches with stride equals 1, generates n scores for each, then retrieves the patches with max scores in each sampling. Then the Dynamic Reassembly generates scores for selected patches, and sorts them based on the scores to determine the sequence. In the Embedding phase, both the embeddings from the Dynamic Reassembly and Conventional Patching are adaptively fused to form the representations.

Experiments

Table 1: Multivariate forecasting average results, SRSNet v.s recent SOTAs.

Datasets	ETTh1		ETTh2		ETTm1		ETTm2		Weather		Electricity		Solar		Traffic	
Metrics	mse	mae	mse	mae	mse	mae	mse	mae	mse	mae	mse	mae	mse	mae	mse	mae
FEDformer [2022]	0.433	0.454	0.406	0.438	0.567	0.519	0.335	0.380	0.312	0.356	0.219	0.330	0.641	0.628	0.620	0.382
Stationary [2022]	0.667	0.568	0.377	0.419	0.531	0.472	0.384	0.390	0.287	0.310	0.194	0.295	0.390	0.387	0.622	0.340
DLinear [2023]	0.430	0.443	0.470	0.468	0.356	0.378	0.259	0.324	0.242	0.295	0.167	0.264	0.224	0.286	0.418	0.287
TimesNet [2023]	0.468	0.459	0.390	0.417	0.408	0.415	0.292	0.331	0.255	0.282	0.190	0.284	0.211	0.281	0.617	0.327
Crossformer [2023]	0.439	0.461	0.894	0.680	0.464	0.456	0.501	0.505	0.232	0.294	0.171	0.263	0.205	**0.232**	0.522	0.282
PatchTST [2023]	0.419	0.436	0.351	0.395	0.349	0.381	0.256	0.314	0.224	**0.262**	0.171	0.270	0.200	0.284	0.397	0.275
TimeMixer [2024]	0.427	0.441	0.347	0.394	0.356	0.380	0.257	0.318	0.225	0.263	0.185	0.284	0.203	0.261	0.410	0.279
iTransformer [2024]	0.440	0.445	0.359	0.396	0.347	0.378	0.258	0.318	0.232	0.270	0.163	0.258	0.202	0.260	0.397	0.281
Amplifier [2025]	0.421	0.433	0.356	0.402	0.353	0.379	0.256	0.318	**0.223**	0.264	0.163	0.256	0.202	0.256	0.417	0.290
TimeKAN [2025]	0.409	0.427	0.350	0.397	**0.344**	0.380	0.260	0.318	0.226	0.268	0.164	0.258	0.198	0.263	0.420	0.286
SRSNet	**0.404**	**0.424**	**0.334**	**0.385**	0.351	**0.378**	**0.252**	**0.314**	0.226	0.266	**0.161**	**0.254**	**0.183**	0.239	**0.392**	**0.270**

- Comprehensive experiments on real-world datasets demonstrate that SRSNet achieves state-of-the-art performance.
- the SRS module can effectively improve the performance of patch-based models.

Table 2: SRS as a plugin.

Datasets	ETTh1		ETTm2		Solar		Traffic	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MLP	0.430	0.451	0.273	0.336	0.219	0.277	0.413	0.284
+ SRS	0.404	0.424	0.252	0.315	0.183	0.239	0.392	0.270
Improve	**6.05%**	**6.06%**	**7.70%**	**6.31%**	**16.08%**	**13.47%**	**5.26%**	**4.88%**
PatchTST	0.419	0.436	0.256	0.314	0.200	0.284	0.397	0.275
+ SRS	0.404	0.426	0.249	0.307	0.182	0.251	0.386	0.266
Improve	**3.48%**	**2.20%**	**3.00%**	**2.33%**	**8.87%**	**11.13%**	**2.74%**	**3.34%**
Crossformer	0.439	0.461	0.501	0.505	0.205	0.232	0.522	0.282
+ SRS	0.432	0.455	0.454	0.462	0.193	0.225	0.512	0.274
Improve	**1.55%**	**1.45%**	**9.83%**	**8.22%**	**5.66%**	**2.92%**	**1.85%**	**2.84%**
PatchMLP	0.435	0.443	0.261	0.322	0.193	0.250	0.413	0.287
+ SRS	0.422	0.436	0.253	0.315	0.179	0.242	0.402	0.277
Improve	**2.99%**	**1.55%**	**2.94%**	**2.31%**	**7.00%**	**3.11%**	**2.57%**	**3.41%**
xPatch	0.416	0.429	0.253	0.308	0.186	0.210	0.398	0.248
+ SRS	0.406	0.422	0.244	0.303	0.179	0.204	0.389	0.240
Improve	**2.38%**	**1.66%**	**3.59%**	**1.86%**	**4.01%**	**2.61%**	**2.17%**	**3.38%**

Efficiency analyses

Table 3: Efficiency analyses of SRSNet and the SRS module on ETTh1 and Solar datasets with look-back length equals 512, forecasting horizon equals 720, and batch size equals 32.The Max GPU Memory (MB), inference Time (s) per batch, Training Time (s) per batch, and MultiplyAccumulate Operations (MACs) are reported as the main metrics.

Datasets		ETTh1		Solar	
Metrics		Memory (MB)	Training Time (s)	Memory (MB)	Training Time (s)
Linear	DLinear	828	1.28	815	15.66
	Amplifier	596	1.78	715	14.38
CNN	TimesNet	2,846	14.95	13,141	1812.46
Transformer	FEDformer	8,190	64.83	3,751	227.45
	Stationary	18,386	40.22	18,529	156.27
	Crossformer	3,976	17.13	16,375	205.60
	PatchTST	1,404	2.49	26,777	137.60
	iTransformer	722	4.14	1,015	20.66
MLP	TimeMixer	1,394	7.49	20,602	107.15
	TimeKAN	1,456	5.50	13,109	326.38
	SRSNet	1,012	2.27	6,301	56.149

Datasets	Variants	Memory (MB)	Inference (s)	Training (s)	MACs (G)
ETTh1	PatchTST	2,837	5.076	5.131	16.214
	+SRS	2,907	5.722	5.763	16.905
	Overhead	+2.47%	+12.73%	+12.31%	+4.26%
	Crossformer	4,011	27.503	32.613	56.280
	+SRS	4,159	30.311	35.276	56.625
	Overhead	+3.69%	+10.21%	+8.17%	+0.61%
Solar	PatchTST	27,822.08	84.231	88.714	600.261
	+SRS	29,767.68	95.200	101.981	613.790
	Overhead	+6.99%	+13.02%	+14.95%	+2.25%
	Crossformer	17,355	79.031	82.472	61.822
	+SRS	18,978	86.674	90.268	62.174
	Overhead	+9.35%	+9.67%	+9.45%	+0.57%

Visualization

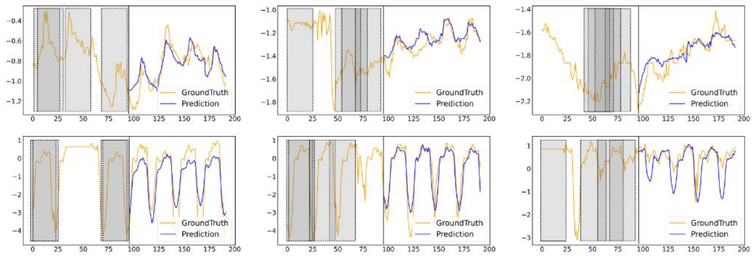

Figure 4: Visualization of input-96-predict-96 results on the ETTh1 dataset. SRSNet effectively processes the special cases with the help of SRS module. The grey rectangles are the selected patches with the size of 24.

My WeChat: