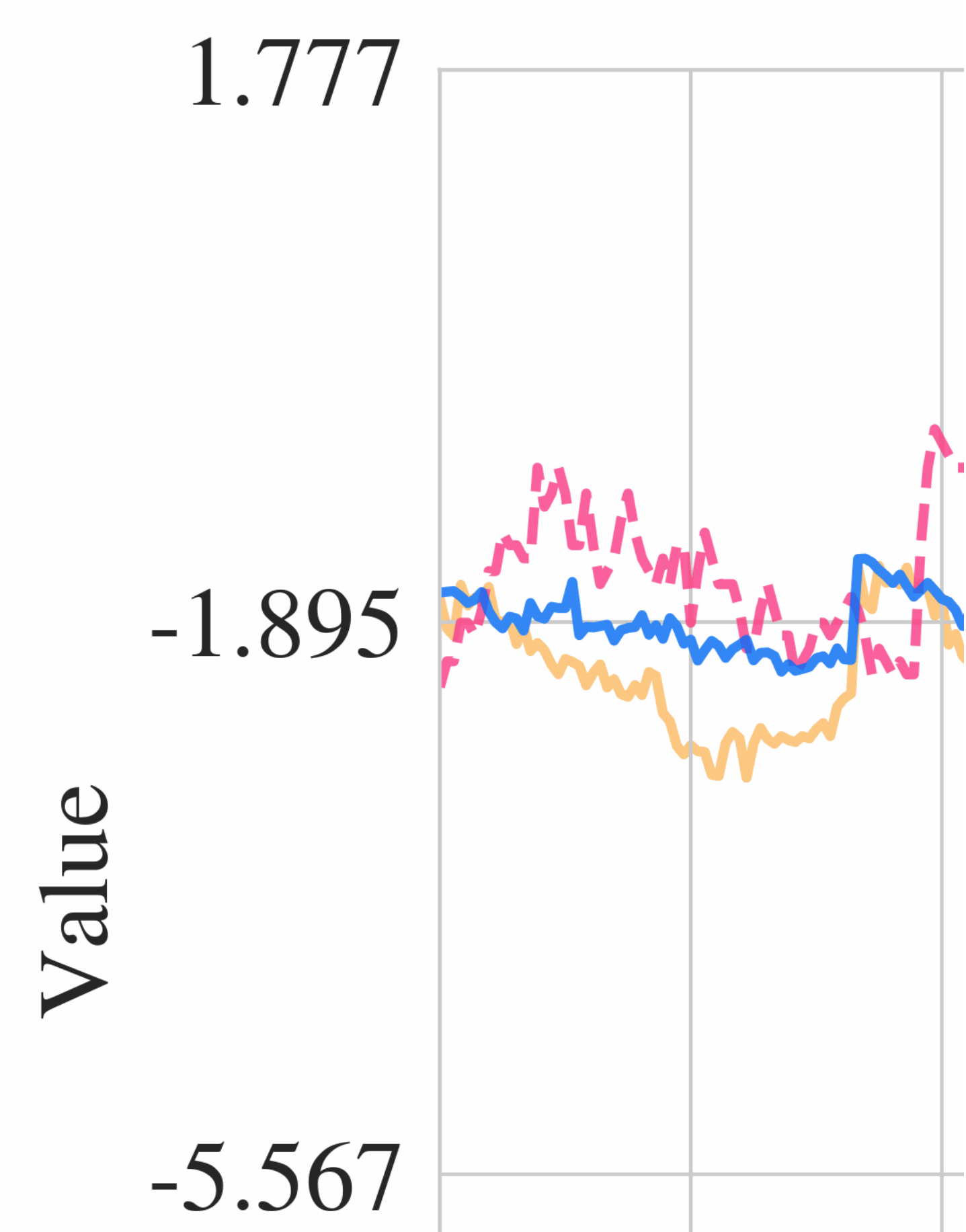
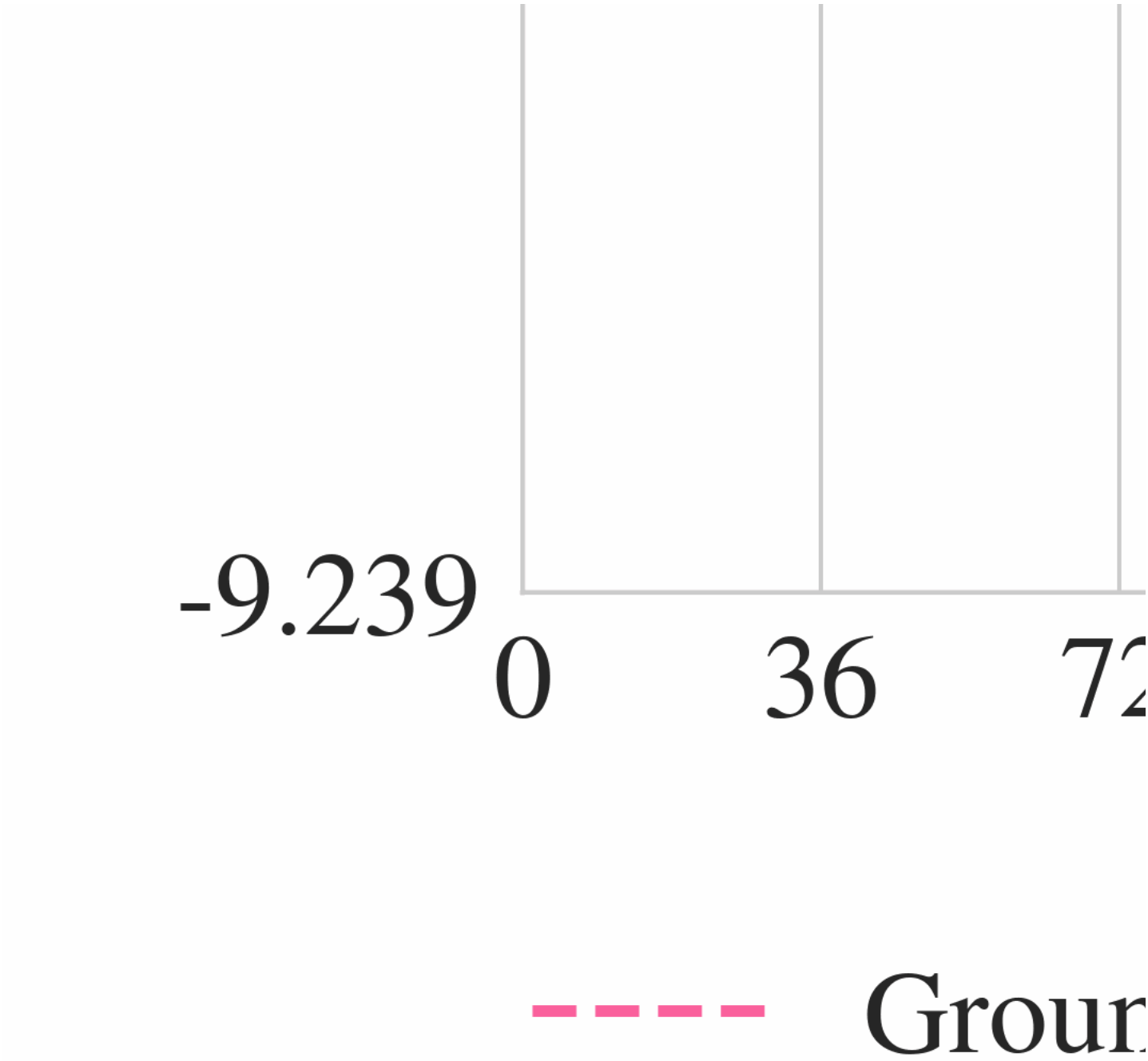
Continuous Evolution Pool: Taming Recurring Concept Drift in Online Time Series Forecasting

Repo Status:





Introduction

Accurate time series forecasting is a fundamental task in various fields such as finance, energy management, traffic prediction, and environmental monitoring. However, online time series forecasting often faces a significant challenge known as concept drift, especially recurring concept drift. Recurring concept drift refers to the periodic reappearance of certain data patterns after a period of absence. Existing solutions mainly rely on parameter updating techniques, which may lead to the loss of previously learned knowledge and lack effective knowledge retention mechanisms.

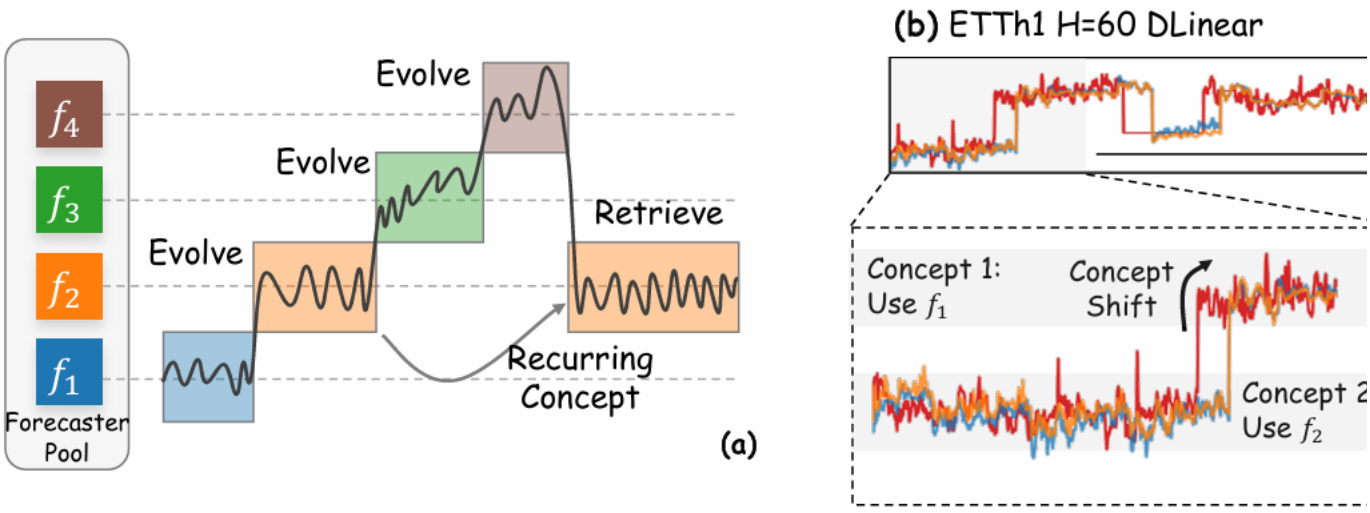


Figure 1: Concept recurrence is a notable phenomenon the second concept recurs following the emergence of th and concept drifts, forecasters often suffer from forgettir To address this challenge and effectively track recurring employs an evolving and retrieval strategy, assigning distinc

The Continuous Evolution Pool (CEP)

To address these limitations, we propose the Continuous Evolution Pool (CEP), a novel pooling mechanism designed to store multiple forecasters corresponding to different concepts. When a new test sample arrives, CEP selects the nearest forecaster in the pool for prediction and learns from the features of its neighboring samples. If there are insufficient neighboring samples, it indicates the emergence of a new concept, and a new model will be added to the pool. Additionally, CEP employs an elimination mechanism to remove outdated knowledge and filter noisy data.

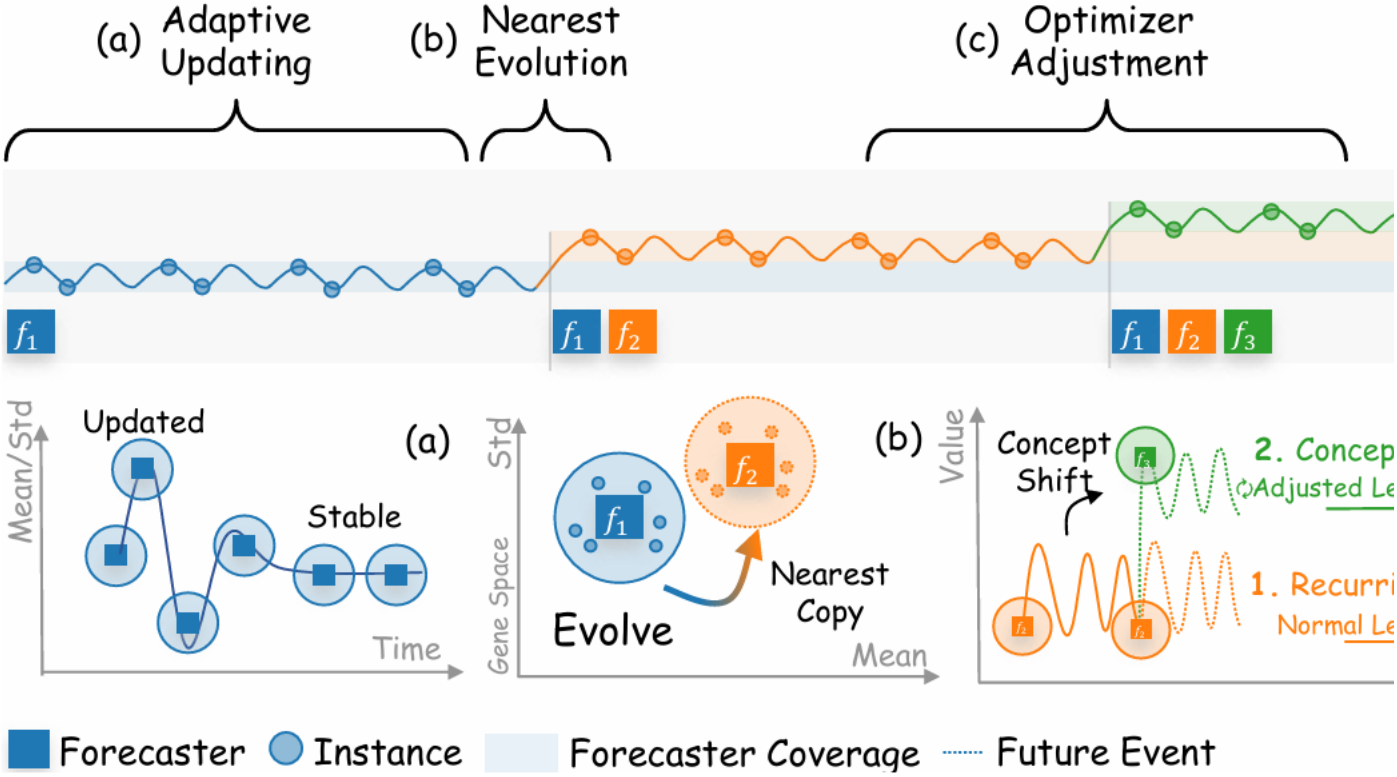


Figure 3: Illustration of the proposed **CEP** mechanism: update the position of the evolved forecaster in the gene sp **Evolution**: If the instance surpasses the forecaster's evol closest forecaster at the instance. (c) **Optimizer Adjustme** concept ■ to ensure accurate adaptation. (d) **Forecaster E** with the rarely-occurring noise concept ■ may be removed **Retrieval**: Identify the nearest forecaster in the gene space

Main Contributions

1. We identify recurring concept drift as a significant challenge in online time series forecasting and point out the shortcomings of existing methods.
2. We propose the CEP mechanism, which effectively mitigates knowledge loss and enables the model to leverage previously acquired knowledge more effectively.
3. Through extensive experiments on multiple real - world datasets and various neural network architectures, we demonstrate that CEP substantially enhances prediction accuracy in scenarios characterized by recurring concept drift.

Main Results

Comparison with Naive Forecasting Models

The experiment results in following table show that the proposed CEP generally reduces the Mean Squared Error (MSE) in forecasting across various datasets and model architectures. This indicates that CEP can effectively capture and utilize features of different concepts, enhancing the performance of online time series forecasting in scenarios with recurring concept drift. Its effectiveness is not limited to a single architecture or dataset, demonstrating broad applicability. Notably, the Crossformer model experiences a significant improvement in univariate online forecasting, with an MSE reduction exceeding 25%. This is due to Crossformer's design that leverages both time and variable dimensions. In contrast, models like PatchTST, which rely solely on the time dimension, may struggle to adapt to complex concept drifts. However, in rare cases, CEP may not achieve the expected improvements, possibly because a single set of hyperparameters was used for all experiments, which may not be optimal for every dataset.

Table 1: The averaged MSE errors of different forecasters when using CEP are prese and reduced outcomes are marked with and respectively; enhancements exc highlighted in . The **best** and second-best performances are highlighted. The presented in Appendix Table 8.

Data	TimeMixer	+CEP	iTransformer	+CEP	PatchTST	+CEP	DLinear	+CEP	SegRNN	+CEP	TimesNet	+CEP	Crossformer
ECL	0.294	0.271	0.275	**0.256**	0.330	0.291	0.307	0.299	0.278	0.274	0.313	0.280	0.530
	-	-7.84%	-	-7.09%	-	-11.97%	-	-2.61%	-	-1.44%	-	-10.70%	-
ETTh1	0.337	0.324	0.328	0.326	0.331	0.323	0.337	0.308	0.304	**0.300**	0.355	0.333	0.636
	-	-3.86%	-	-0.61%	-	-2.27%	-	-8.75%	-	-1.48%	-	-6.20%	-
ETTh2	2.713	2.526	2.547	2.518	2.667	**2.467**	2.717	2.630	2.629	2.584	2.642	2.564	7.743
	-	-6.91%	-	-1.14%	-	-7.52%	-	-3.22%	-	-1.73%	-	-2.95%	-
ETTm1	0.778	0.747	0.777	0.738	0.762	0.720	0.849	0.751	0.718	**0.715**	0.818	0.820	2.439
	-	-3.92%	-	-5.02%	-	-5.52%	-	-11.54%	-	-0.49%	-	0.31%	-
ETTm2	**0.231**	0.232	0.241	0.236	0.255	0.245	0.242	0.234	0.233	0.227	0.248	0.241	0.416
	-	0.43%	-	-2.07%	-	-4.12%	-	-3.31%	-	-2.58%	-	-2.82%	-
Exchange	0.358	0.344	0.376	0.361	0.354	0.346	0.412	0.381	0.296	**0.294**	0.470	0.393	2.168
	-	-3.91%	-	-3.86%	-	-2.26%	-	-7.53%	-	-0.68%	-	-16.40%	-
Traffic	0.512	0.500	0.521	**0.492**	0.677	0.587	0.628	0.628	0.795	0.783	0.559	0.563	0.668
	-	-2.44%	-	-5.48%	-	-13.30%	-	0.00%	-	-1.51%	-	0.72%	-
WTH	0.359	0.356	0.355	0.354	0.367	0.353	0.342	**0.341**	0.342	0.342	0.370	0.376	0.399
	-	-0.84%	-	-0.42%	-	-3.81%	-	-0.29%	-	-0.15%	-	1.49%	-

Comparison with Online Forecasting Models

Results in following table reveal that ER, DER++, FSNet, and OneNet did not perform well in delayed scenarios. Experience replay methods reuse instances from memory to update the model, but limited memory capacity may cause them to lose earlier concepts due to concept drift. FSNet is prone to forecasting biases due to delayed sample arrival, while OneNet adjusts model parameters using short - term information to optimize forecasts in delayed scenarios to some extent. To comprehensively preserve knowledge of different concepts, CEP uses a pooling mechanism to assign different concepts to distinct models. In delay - feedback scenarios, CEP significantly outperforms previous time series methods, reducing the MSE by over 20%. This suggests that previous online methods are less effective in delay scenarios with recurring concepts and may even be inferior to the naive model.

Table 2: The average MSE error of previc all integrated TCN according to the origin and reduced outcomes are marked with highlighted in . The **best** and second-best in Appendix Table 9.

Data	ECL	ETTh1	ETTh2
TCN	0.355	0.454	3.201
ER	0.636 79.41%	0.498 9.58%	3.346 4.53%
DER++	0.588 65.73%	0.461 1.54%	3.150 -1.61%
FSNet	0.863 143.30%	0.452 -0.44%	3.800 18.70%
OneNet	0.401 13.12%	**0.439** -3.30%	3.505 9.50%
CEP	**0.316** -11.00%	0.445 -2.09%	**2.811** -12.18%

Visualization Comparison

Figure 6 shows the forecasting results of the naive model and CEP. After the concept recurred (instance 2 - 14), CEP's MSE remained stable in Figure 7, while the naive model's MSE rose significantly. Figure 8 visualizes CEP's operation, with forecasts of different concepts marked by distinct colors,

demonstrating CEP's ability to quickly and accurately identify concepts.

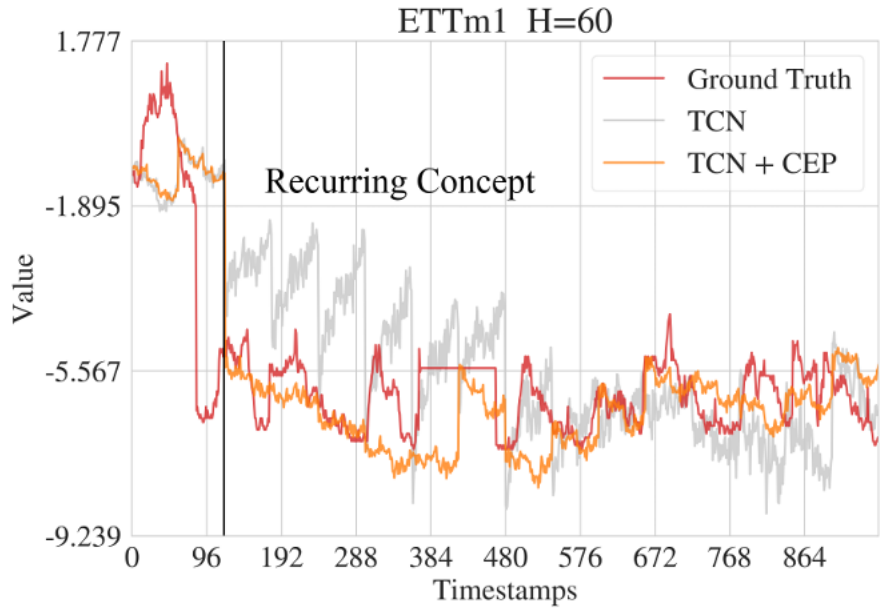


Figure 6: Visualization of forecasting results

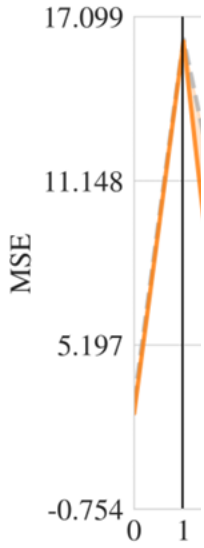


Figure 7
ric

Installation

1. Clone the repository:
2. Install the required dependencies:

```
pip install -r requirements.txt
```

Usage

Extract the Dataset

```
unzip ./data/data.zip -d ./data/
```

Manually Running Experiments

1. Run the main script:

```
python main.py --some_parameter XXX
```

Running Experiments with Configurations

1. Run the main script with a configuration file:

```
python main.py --config ./configs/main/ep_cep_da_ECL_ml_CEP_fr_Crossformer_pn_1_fo_0.8_do_1.5_oe_fade_pt_True.json
```

Reproducing Experiments from the Paper

1. Main Experiment: Run the script in `./run_scripts`:

```
python run_scripts/cep.py
```

2. Ablation Study: To verify CEP components effectiveness:

```
python run_scripts/ablation.py
```

3. Baseline Comparison: Reproduce baseline methods results:

```
python run_scripts/baseline.py
```

4. Parameter Sensitivity: Test key parameters' impacts:

```
python run_scripts/parameter_sensitivity.py
```

Repository Structure

- `configs`: Contains configuration files for different experiments.
- `data`: Includes data loading scripts and datasets.
- `exp`: Contains experiment scripts.
- `layers`: Contains neural network layer implementations.
- `models`: Contains different model implementations.
- `utils`: Contains utility functions.

Citation

If you find this repo useful, please cite our paper.

```
@misc{zhan2025continuousevolutionpooltaming,
      title={Continuous Evolution Pool: Taming Recurring Concept Drift in Online Time Series Forecasting},
      author={Tianxiang Zhan and Ming Jin and Yuanpeng He and Yuxuan Liang and Yong Deng and Shirui Pan},
      year={2025},
      eprint={2506.14790},
      archivePrefix={arXiv},
      primaryClass={cs.LG},
      url={https://arxiv.org/abs/2506.14790},
}
```

License

This source code is released under the MIT license, included here.

Acknowledgement

This library is constructed based on the following repos:

- FSNet https://github.com/salesforce/fsnet/.
- OneNet: https://github.com/yfzhang114/OneNet/.
- Time Series Library: https://github.com/thuml/Time-Series-Library/.

Star History