Informer 모델과 DLinear 모델의 장기 시계열 예측 성능 비교

데이터사이언스개론 기말 프로젝트 (3조)

아주대학교 소프트웨어융합대학 미디어학과 박성범

2022년 12월 7일

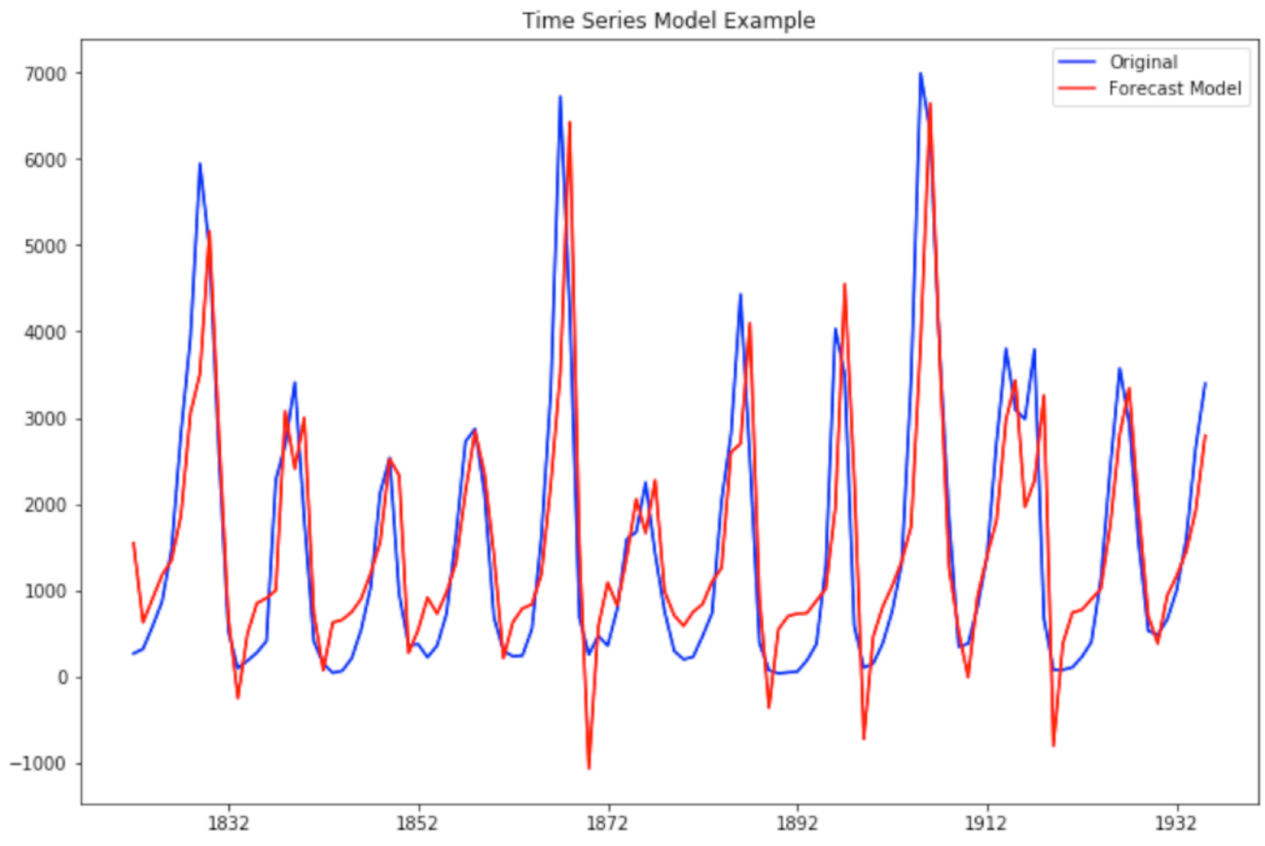
목차

용어

- **시계열**: 일정 시간 간격으로 배치된 데이터의 수열.
  (e.g. 특정 기간 내의 교통량, 날짜별 기온 등)

- **시계열 예측**: 지난 시계열 데이터를 바탕으로 미래의 시계열 데이터를 예측.

- **장기 시계열 예측**:
  49건 이상의 시점 예측.



1. 주제

- Informer 모델과 DLinear 모델의
  장기 시계열 예측 성능 비교.

- MSE, MAE, 학습 시간, 테스트 시간을 지표로 사용.

- Informer (H. Zhou et al., 2021):
  Transformer 기반 장기 시계열 예측 모델.

- DLinear (A. Zeng et al., 2022):
  선형 모델 기반 장기 시계열 예측 모델.
  Transformer 기반 모델에 대한 의문 제기.

- **DLinear 논문을 재현함으로써 주장의 유효성을 검증.**

2. 배경

- 전통적 통계학 기반: ARIMA, Prophet(2017)
- Decision Tree 기반: GBRT
- RNN 기반: LSTM(1997), GRU
- CNN 기반: TCN(2018), SCINet(2021)
- Transformer 기반: **Informer(Mar 2021)**
- 선형 모델 기반: **DLinear(May 2022)**

2.A. 배경: Informer

- H. Zhou et al., “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting”, 2021.
- Transformer 기반 장기 시계열 예측 모델.
- AAAI 2021 Best Paper
- 2021 ETTh1 SOTA

Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting

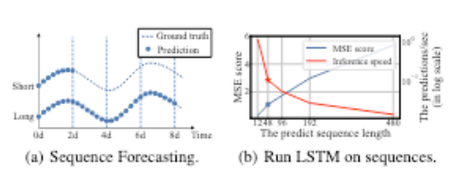
Haoyi Zhou, [1] **Shanghang Zhang,** [2] **Jieqi Peng,** [1] **Shuai Zhang,** [1] **Jianxin Li,** [1]
Hui Xiong, [3] **Wancai Zhang** [4]
[1] Beihang University [2] UC Berkeley [3] Rutgers University [4] SEDD Company
{zhouhy, pengjq, zhangs, lijx}@act.buaa.edu.cn, shz@eecs.berkeley.edu, {xionghui,zhangwancaibuaa}@gmail.com

Abstract

Many real-world applications require the prediction of long sequence time-series, such as electricity consumption planning. Long sequence time-series forecasting (LSTF) demands a high prediction capacity of the model, which is the ability to capture precise long-range dependency coupling between output and input efficiently. Recent studies have shown the potential of Transformer to increase the prediction capacity. However, there are several severe issues with Transformer that prevent it from being directly applicable to LSTF, including quadratic time complexity, high memory usage, and inherent limitation of the encoder-decoder architecture. To address these issues, we design an efficient transformer-based model for LSTF, named Informer, with three distinctive characteristics: (i) a *ProbSparse* self-attention mechanism, which achieves $\mathcal{O}(L \log L)$ in time complexity and memory usage, and has comparable performance on sequences' dependency alignment. (ii) the self-attention distilling highlights dominating attention by halving cascading layer input, and efficiently handles extreme long input sequences. (iii) the generative style decoder, while conceptually simple, predicts the long time-series sequences at one forward operation rather than a step-by-step way, which drastically improves the inference speed of long-sequence predictions. Extensive experiments on four large-scale datasets demonstrate that Informer significantly outperforms existing methods and provides a new solution to the LSTF problem.

1 Introduction

Time-series forecasting is a critical ingredient across many domains, such as sensor network monitoring (Papadimitriou and Yu, 2006), energy and smart grid management, economics and finance (Zhu and Shasha, 2002), and disease propagation analysis (Matsubara et al., 2014). In these scenarios, we can leverage a substantial amount of time-series data on past behavior to make a forecast in the long run, namely long sequence time-series forecasting (LSTF). However, existing methods are mostly designed under short-term problem setting, like predicting 48 points or less (Hochreiter and Schmidhuber, 1997; Li et al., 2018; Yu et al., 2017; Liu et al., 2019; Qin et al., 2017; Wen et al., 2017). The increasingly long sequences strain the models' prediction capacity to the point where this trend is holding the research on LSTF. As an empirical example, Fig.(1) shows the forecasting results on a real dataset, where the LSTM network predicts the



(a) Sequence Forecasting. (b) Run LSTM on sequences.

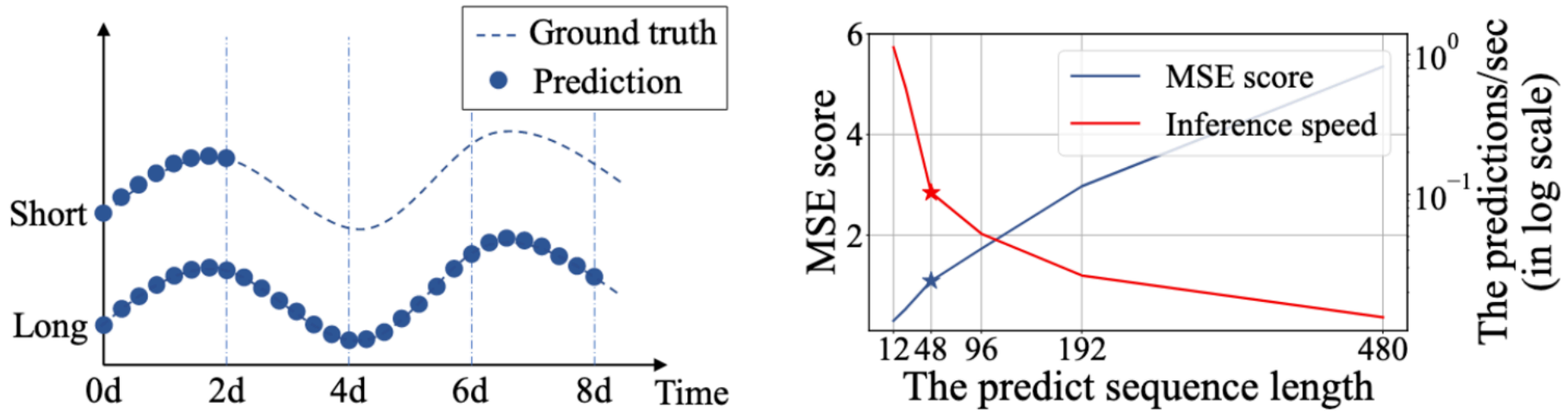
Figure 1: (a) LSTF can cover an extended period than the short sequence predictions, making vital distinction in policy-planning and investment-protecting. (b) The prediction capacity of existing methods limits LSTF's performance. E.g., starting from length=48, MSE rises unacceptably high, and the inference speed drops rapidly.

hourly temperature of an electrical transformer station from the short-term period (12 points, 0.5 days) to the long-term period (480 points, 20 days). The overall performance gap is substantial when the prediction length is greater than 48 points (the solid star in Fig.(1b)), where the MSE rises to unsatisfactory performance, the inference speed gets sharp drop, and the LSTM model starts to fail.

The major challenge for LSTF is to enhance the prediction capacity to meet the increasingly long sequence demand, which requires (a) extraordinary long-range alignment ability and (b) efficient operations on long sequence inputs and outputs. Recently, Transformer models have shown superior performance in capturing long-range dependency than RNN models. The self-attention mechanism can reduce the maximum length of network signals traveling paths into the theoretical shortest $\mathcal{O}(1)$ and avoid the recurrent structure, whereby Transformer shows great potential for the LSTF problem. Nevertheless, the self-attention mechanism violates requirement (b) due to its L-quadratic computation and memory consumption on L-length inputs/outputs. Some large-scale Transformer models pour resources and yield impressive results on NLP tasks (Brown et al., 2020), but the training on dozens of GPUs and expensive deploying cost make theses models unaffordable on real-world LSTF problem. The efficiency of the self-attention mechanism and Transformer architecture becomes the bottleneck of applying them to LSTF problems. Thus, in this paper, we seek to answer the question: *can we improve Transformer models to*

2.A. 배경: Informer

- 기존 시계열 예측 모델은 모두 단기 문제에만 적합.



(a) Sequence Forecasting. (b) Run LSTM on sequences.

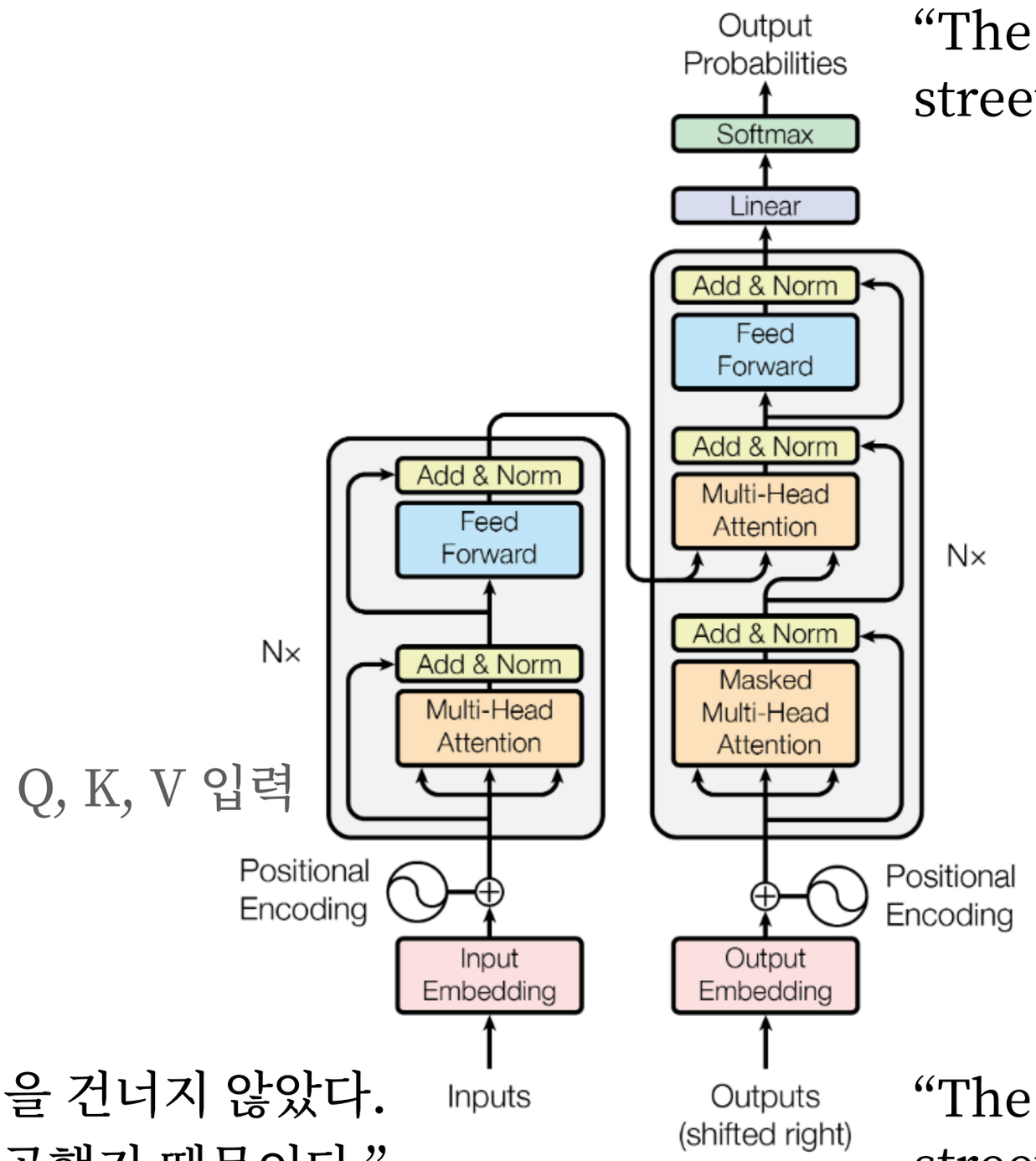
- Transformer 모델은 RNN 모델에 비해 높은 성능을 달성.
- 특정 시점을 비교할 때 $O(1)$ 시간만 소요.
- 하지만 **self-attention** 매커니즘으로 $O(L^2)$ 시간이 소요.

2.A. 배경: Informer: Transformer

- **A. Vaswani et al., "Attention Is All You Need", 2017.**
- 문장 속 단어와 같은 순차 데이터 내의 관계를
  추적해 맥락과 의미를 학습하는 모델.
- **Self-Attention**:
  입력 시퀀스를 구성하는 단어 사이의 관계를 파악.
  - "**남자**는 길을 건너지 않았다. **그**는 피곤했기 때문이다"

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

2.A. 배경: Informer: Transformer



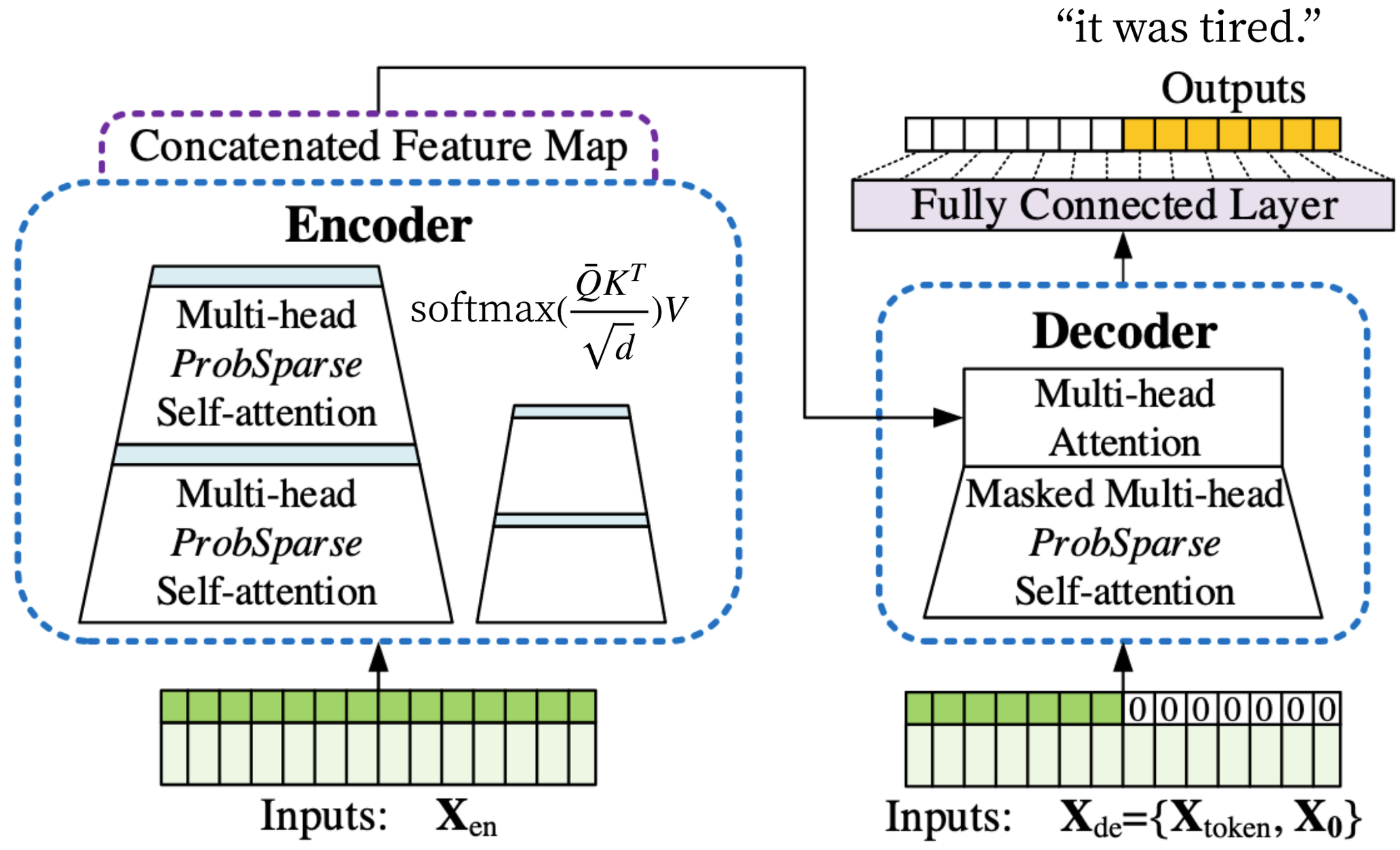
"The man didn't cross the street. Because ..."

Q, K, V 입력

"남자는 길을 건너지 않았다.
피곤했기 때문이다."

"The man didn't cross the street. Because it was tired."

2.A. 배경: Informer



"남자는 길을 건너지 않았다.
피곤했기 때문이다."

"The man didn't cross the street. Because 000 … 000"

2.A. 배경: Informer

Methods		Informer		Informer†		LogTrans		Reformer		LSTMa		DeepAR		ARIMA		Prophet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	0.098	0.247	**0.092**	**0.246**	0.103	0.259	0.222	0.389	0.114	0.272	0.107	0.280	0.108	0.284	0.115	0.275
	48	**0.158**	**0.319**	0.161	0.322	0.167	0.328	0.284	0.445	0.193	0.358	0.162	0.327	0.175	0.424	0.168	0.330
	168	**0.183**	**0.346**	0.187	0.355	0.207	0.375	1.522	1.191	0.236	0.392	0.239	0.422	0.396	0.504	1.224	0.763
	336	0.222	0.387	**0.215**	**0.369**	0.230	0.398	1.860	1.124	0.590	0.698	0.445	0.552	0.468	0.593	1.549	1.820
	720	0.269	0.435	**0.257**	**0.421**	0.273	0.463	2.112	1.436	0.683	0.768	0.658	0.707	0.659	0.766	2.735	3.253
$ETTh_2$	24	**0.093**	**0.240**	0.099	0.241	0.102	0.255	0.263	0.437	0.155	0.307	0.098	0.263	3.554	0.445	0.199	0.381
	48	**0.155**	**0.314**	0.159	0.317	0.169	0.348	0.458	0.545	0.190	0.348	0.163	0.341	3.190	0.474	0.304	0.462
	168	**0.232**	**0.389**	0.235	0.390	0.246	0.422	1.029	0.879	0.385	0.514	0.255	0.414	2.800	0.595	2.145	1.068
	336	0.263	**0.417**	**0.258**	0.423	0.267	0.437	1.668	1.228	0.558	0.606	0.604	0.607	2.753	0.738	2.096	2.543
	720	**0.277**	**0.431**	0.285	0.442	0.303	0.493	2.030	1.721	0.640	0.681	0.429	0.580	2.878	1.044	3.355	4.664
$ETTm_1$	24	**0.030**	**0.137**	0.034	0.160	0.065	0.202	0.095	0.228	0.121	0.233	0.091	0.243	0.090	0.206	0.120	0.290
	48	0.069	0.203	**0.066**	**0.194**	0.078	0.220	0.249	0.390	0.305	0.411	0.219	0.362	0.179	0.306	0.133	0.305
	96	0.194	**0.372**	**0.187**	0.384	0.199	0.386	0.920	0.767	0.287	0.420	0.364	0.496	0.272	0.399	0.194	0.396
	288	**0.401**	0.554	0.409	**0.548**	0.411	0.572	1.108	1.245	0.524	0.584	0.948	0.795	0.462	0.558	0.452	0.574
	672	**0.512**	**0.644**	0.519	0.665	0.598	0.702	1.793	1.528	1.064	0.873	2.437	1.352	0.639	0.697	2.747	1.174
Weather	24	**0.117**	**0.251**	0.119	0.256	0.136	0.279	0.231	0.401	0.131	0.254	0.128	0.274	0.219	0.355	0.302	0.433
	48	**0.178**	0.318	0.185	**0.316**	0.206	0.356	0.328	0.423	0.190	0.334	0.203	0.353	0.273	0.409	0.445	0.536
	168	**0.266**	**0.398**	0.269	0.404	0.309	0.439	0.654	0.634	0.341	0.448	0.293	0.451	0.503	0.599	2.441	1.142
	336	**0.297**	**0.416**	0.310	0.422	0.359	0.484	1.792	1.093	0.456	0.554	0.585	0.644	0.728	0.730	1.987	2.468
	720	**0.359**	**0.466**	0.361	0.471	0.388	0.499	2.087	1.534	0.866	0.809	0.499	0.596	1.062	0.943	3.859	1.144
ECL	48	0.239	0.359	0.238	0.368	0.280	0.429	0.971	0.884	0.493	0.539	**0.204**	**0.357**	0.879	0.764	0.524	0.595
	168	0.447	0.503	0.442	0.514	0.454	0.529	1.671	1.587	0.723	0.655	**0.315**	**0.436**	1.032	0.833	2.725	1.273
	336	0.489	0.528	0.501	0.552	0.514	0.563	3.528	2.196	1.212	0.898	**0.414**	**0.519**	1.136	0.876	2.246	3.077
	720	**0.540**	**0.571**	0.543	0.578	0.558	0.609	4.891	4.047	1.511	0.966	0.563	0.595	1.251	0.933	4.243	1.415
	960	**0.582**	**0.608**	0.594	0.638	0.624	0.645	7.019	5.105	1.545	1.006	0.657	0.683	1.370	0.982	6.901	4.264
Count		32		12		0		0		0		6		0		0	

2.B. 배경: DLinear

- A. Zeng et al., “Are Transformers Effective for Time Series Forecasting?”, 2022.

- 단순한 선형 모델 기반 장기 시계열 예측 모델.

- 시계열 예측에서 트랜스포머 기반 모델에 의문 제기.

Are Transformers Effective for Time Series Forecasting?

Ailing Zeng[1]*, Muxi Chen[1]*, Lei Zhang[2], Qiang Xu[1]
[1]The Chinese University of Hong Kong
[2]International Digital Economy Academy (IDEA)
{alzeng, mxchen21, qxu}@cse.cuhk.edu.hk
{leizhang}@idea.edu.cn

Abstract

Recently, there has been a surge of Transformer-based solutions for the long-term time series forecasting (LTSF) task. Despite the growing performance over the past few years, we question the validity of this line of research in this work. *Specifically, Transformers is arguably the most successful solution to extract the semantic correlations among the elements in a long sequence. However, in time series modeling, we are to extract the temporal relations in* an ordered set of continuous points. *While employing positional encoding and using tokens to embed sub-series in Transformers facilitate preserving some ordering information, the nature of the* permutation-invariant *self-attention mechanism inevitably results in temporal information loss.*

To validate our claim, we introduce a set of embarrassingly simple one-layer linear models named LTSF-Linear *for comparison. Experimental results on nine real-life datasets show that* LTSF-Linear *surprisingly outperforms existing sophisticated Transformer-based LTSF models in all cases, and often by a large margin. Moreover, we conduct comprehensive empirical studies to explore the impacts of various design elements of LTSF models on their temporal relation extraction capability. We hope this surprising finding opens up new research directions for the LTSF task. We also advocate revisiting the validity of Transformer-based solutions for other time series analysis tasks (e.g., anomaly detection) in the future. Code is available at:* https://github.com/cure-lab/LTSF-Linear.

1. Introduction

Time series are ubiquitous in today’s data-driven world. Given historical data, time series forecasting (TSF) is a long-standing task that has a wide range of applications, including but not limited to traffic flow estimation, energy management, and financial investment. Over the past several decades, TSF solutions have undergone a progression from traditional statistical methods (e.g., ARIMA [1]) and machine learning techniques (e.g., GBRT [11]) to deep learning-based solutions, e.g., Recurrent Neural Networks [15] and Temporal Convolutional Networks [3, 17].

Transformer [26] is arguably the most successful sequence modeling architecture, demonstrating unparalleled performances in various applications, such as natural language processing (NLP) [7], speech recognition [8], and computer vision [19, 29]. Recently, there has also been a surge of Transformer-based solutions for time series analysis, as surveyed in [27]. Most notable models, which focus on the less explored and challenging long-term time series forecasting (LTSF) problem, include LogTrans [16] (NeurIPS 2019), Informer [30] (AAAI 2021 Best paper), Autoformer [28] (NeurIPS 2021), Pyraformer [18] (ICLR 2022 Oral), Triformer [5] (IJCAI 2022) and the recent FEDformer [31] (ICML 2022).

The main working power of Transformers is from its multi-head self-attention mechanism, which has a remarkable capability of extracting semantic correlations among elements in a long sequence (e.g., words in texts or 2D patches in images). However, self-attention is *permutation-invariant* and “anti-order” to some extent. While using various types of positional encoding techniques can preserve some ordering information, it is still inevitable to have temporal information loss after applying self-attention on top of them. This is usually not a serious concern for semantic-rich applications such as NLP, e.g., the semantic meaning of a sentence is largely preserved even if we reorder some words in it. However, when analyzing time series data, there is usually a lack of semantics in the numerical data itself, and we are mainly interested in modeling the temporal changes among *a continuous set of points*. That is, the order itself plays the most crucial role. Consequently, we pose the following intriguing question: ***Are Transformers really effective for long-term time series forecasting?***

Moreover, while existing Transformer-based LTSF so-

*Equal contribution

2.B. 배경: DLinear

- Informer 모델이 비교적 높은 성능을 달성했지만,
  해당 논문에서 비교한 Non-Transformer 모델들은
  성능이 안 좋은 것이 당연하다.

- Transformer 모델의 Multi-Head Attention은
  시퀀스 속의 연관성을 파악하는 데는 적절하지만,
  데이터의 시간 순서를 고려하지 않는다.

- 그러나 시계열 예측에는 각 데이터간의 연관성 보다는
  입력 데이터의 시간 순서 자체가 매우 중요하다.

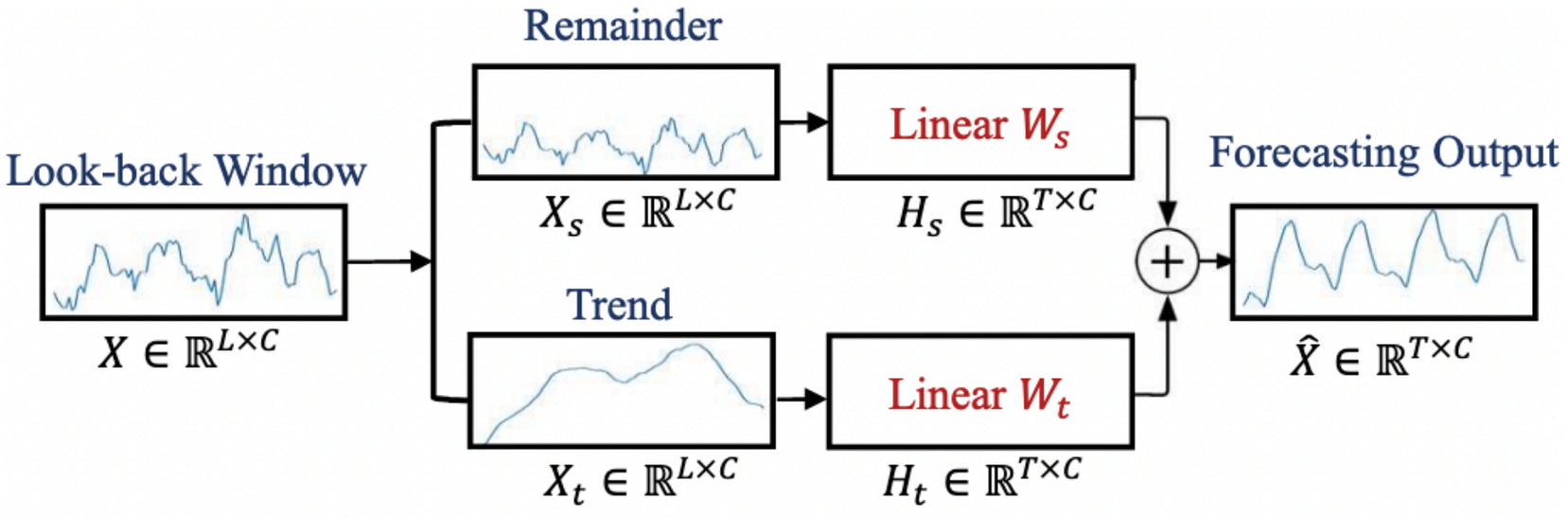
- **정말 Transformer 모델이 장기 시계열 예측에 적합한가?**

2.B. 배경: DLinear

- Self-Attension은 본질적으로 시계열 예측에서 loss를 증가시키는 악효과를 일으킨다.

- **1-Layer 선형 네트워크는 미래를 예측하기 위해 과거의 데이터를 압축할 수 있는 가장 간단한 모델.**

2.B. 배경: DLinear

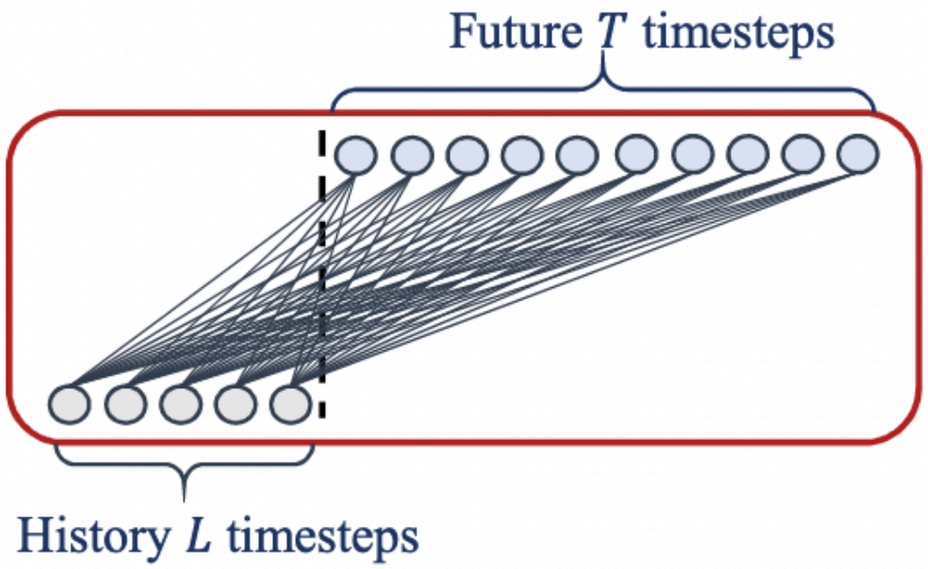


(a) The whole structure of *DLinear*

$$\hat{X} = H_s + H_t$$

Remainder $H_s = W_s X_s$

Trend $H_t = W_t X_t$



(b) One Linear Layer

2.B. 배경: DLinear

Methods		IMP.	Linear*		NLinear*		DLinear*		FEDformer		Autoformer		Informer		Pyraformer*		LogTrans		*Repeat**	
Metric		MSE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	27.40%	**0.140**	**0.237**	0.141	**0.237**	**0.140**	**0.237**	0.193	0.308	0.201	0.317	0.274	0.368	0.386	0.449	0.258	0.357	1.588	0.946
	192	23.88%	**0.153**	0.250	0.154	**0.248**	**0.153**	0.249	0.201	0.315	0.222	0.334	0.296	0.386	0.386	0.443	0.266	0.368	1.595	0.950
	336	21.02%	**0.169**	0.268	0.171	**0.265**	**0.169**	0.267	0.214	0.329	0.231	0.338	0.300	0.394	0.378	0.443	0.280	0.380	1.617	0.961
	720	17.47%	**0.203**	0.301	0.210	**0.297**	**0.203**	0.301	0.246	0.355	0.254	0.361	0.373	0.439	0.376	0.445	0.283	0.376	1.647	0.975
Exchange	96	45.27%	0.082	0.207	0.089	0.208	**0.081**	0.203	0.148	0.278	0.197	0.323	0.847	0.752	0.376	1.105	0.968	0.812	**0.081**	**0.196**
	192	42.06%	0.167	0.304	0.180	0.300	**0.157**	0.293	0.271	0.380	0.300	0.369	1.204	0.895	1.748	1.151	1.040	0.851	0.167	**0.289**
	336	33.69%	0.328	0.432	0.331	0.415	**0.305**	0.414	0.460	0.500	0.509	0.524	1.672	1.036	1.874	1.172	1.659	1.081	**0.305**	**0.396**
	720	46.19%	0.964	0.750	1.033	0.780	**0.643**	**0.601**	1.195	0.841	1.447	0.941	2.478	1.310	1.943	1.206	1.941	1.127	0.823	0.681
Traffic	96	30.15%	**0.410**	0.282	**0.410**	**0.279**	**0.410**	0.282	0.587	0.366	0.613	0.388	0.719	0.391	2.085	0.468	0.684	0.384	2.723	1.079
	192	29.96%	**0.423**	0.287	**0.423**	**0.284**	**0.423**	0.287	0.604	0.373	0.616	0.382	0.696	0.379	0.867	0.467	0.685	0.390	2.756	1.087
	336	29.95%	0.436	0.295	**0.435**	**0.290**	0.436	0.296	0.621	0.383	0.622	0.337	0.777	0.420	0.869	0.469	0.734	0.408	2.791	1.095
	720	25.87%	0.466	0.315	**0.464**	**0.307**	0.466	0.315	0.626	0.382	0.660	0.408	0.864	0.472	0.881	0.473	0.717	0.396	2.811	1.097
Weather	96	18.89%	**0.176**	0.236	0.182	**0.232**	**0.176**	0.237	0.217	0.296	0.266	0.336	0.300	0.384	0.896	0.556	0.458	0.490	0.259	0.254
	192	21.01%	**0.218**	0.276	0.225	**0.269**	0.220	0.282	0.276	0.336	0.307	0.367	0.598	0.544	0.622	0.624	0.658	0.589	0.309	0.292
	336	22.71%	**0.262**	0.312	0.271	**0.301**	0.265	0.319	0.339	0.380	0.359	0.395	0.578	0.523	0.739	0.753	0.797	0.652	0.377	0.338
	720	19.85%	0.326	0.365	0.338	**0.348**	**0.323**	0.362	0.403	0.428	0.419	0.428	1.059	0.741	1.004	0.934	0.869	0.675	0.465	0.394
ILI	24	47.86%	1.947	0.985	**1.683**	**0.858**	2.215	1.081	3.228	1.260	3.483	1.287	5.764	1.677	1.420	2.012	4.480	1.444	6.587	1.701
	36	36.43%	2.182	1.036	**1.703**	**0.859**	1.963	0.963	2.679	1.080	3.103	1.148	4.755	1.467	7.394	2.031	4.799	1.467	7.130	1.884
	48	34.43%	2.256	1.060	**1.719**	**0.884**	2.130	1.024	2.622	1.078	2.669	1.085	4.763	1.469	7.551	2.057	4.800	1.468	6.575	1.798
	60	34.33%	2.390	1.104	**1.819**	**0.917**	2.368	1.096	2.857	1.157	2.770	1.125	5.264	1.564	7.662	2.100	5.278	1.560	5.893	1.677
ETTh1	96	0.80%	0.375	0.397	**0.374**	**0.394**	0.375	0.399	0.376	0.419	0.449	0.459	0.865	0.713	0.664	0.612	0.878	0.740	1.295	0.713
	192	3.57%	0.418	0.429	0.408	**0.415**	**0.405**	0.416	0.420	0.448	0.500	0.482	1.008	0.792	0.790	0.681	1.037	0.824	1.325	0.733
	336	6.54%	0.479	0.476	**0.429**	**0.427**	0.439	0.443	0.459	0.465	0.521	0.496	1.107	0.809	0.891	0.738	1.238	0.932	1.323	0.744
	720	13.04%	0.624	0.592	**0.440**	**0.453**	0.472	0.490	0.506	0.507	0.514	0.512	1.181	0.865	0.963	0.782	1.135	0.852	1.339	0.756
ETTh2	96	19.94%	0.288	0.352	**0.277**	**0.338**	0.289	0.353	0.346	0.388	0.358	0.397	3.755	1.525	0.645	0.597	2.116	1.197	0.432	0.422
	192	19.81%	0.377	0.413	**0.344**	**0.381**	0.383	0.418	0.429	0.439	0.456	0.452	5.602	1.931	0.788	0.683	4.315	1.635	0.534	0.473
	336	25.93%	0.452	0.461	**0.357**	**0.400**	0.448	0.465	0.496	0.487	0.482	0.486	4.721	1.835	0.907	0.747	1.124	1.604	0.591	0.508
	720	14.25%	0.698	0.595	**0.394**	**0.436**	0.605	0.551	0.463	0.474	0.515	0.511	3.647	1.625	0.963	0.783	3.188	1.540	0.588	0.517
ETTm1	96	21.10%	0.308	0.352	0.306	0.348	**0.299**	**0.343**	0.379	0.419	0.505	0.475	0.672	0.571	0.543	0.510	0.600	0.546	1.214	0.665
	192	21.36%	0.340	0.369	0.349	0.375	**0.335**	**0.365**	0.426	0.441	0.553	0.496	0.795	0.669	0.557	0.537	0.837	0.700	1.261	0.690
	336	17.07%	0.376	0.393	0.375	0.388	**0.369**	**0.386**	0.445	0.459	0.621	0.537	1.212	0.871	0.754	0.655	1.124	0.832	1.283	0.707
	720	21.73%	0.440	0.435	0.433	0.422	**0.425**	**0.421**	0.543	0.490	0.671	0.561	1.166	0.823	0.908	0.724	1.153	0.820	1.319	0.729
ETTm2	96	17.73%	0.168	0.262	**0.167**	**0.255**	**0.167**	0.260	0.203	0.287	0.255	0.339	0.365	0.453	0.435	0.507	0.768	0.642	0.266	0.328
	192	17.84%	0.232	0.308	**0.221**	**0.293**	0.224	0.303	0.269	0.328	0.281	0.340	0.533	0.563	0.730	0.673	0.989	0.757	0.340	0.371
	336	15.69%	0.320	0.373	**0.274**	**0.327**	0.281	0.342	0.325	0.366	0.339	0.372	1.363	0.887	1.201	0.845	1.334	0.872	0.412	0.410
	720	12.58%	0.413	0.435	**0.368**	**0.384**	0.397	0.421	0.421	0.415	0.433	0.432	3.379	1.338	3.625	1.451	3.048	1.328	0.521	0.465

- Methods* are implemented by us; Other results are from FEDformer [31].

3. 비교 분석

- 두 모델에 대해 ETTh1 데이터셋을 대상으로 한 예측 테스트 결과를 재현한다.
- 336건의 과거 데이터를 바탕으로 96건의 미래를 예측.
- MSE, MAE 지표.
- 학습/테스트 소요 시간.
- Google Colab: Python 3 Google Computer Engine 백엔드 (GPU)

```
1 !python --version
```

```
Python 3.8.15
```

```
1 !nvidia-smi
```

```
Wed Dec  7 08:30:27 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   45C    P0    27W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

3. 비교 분석

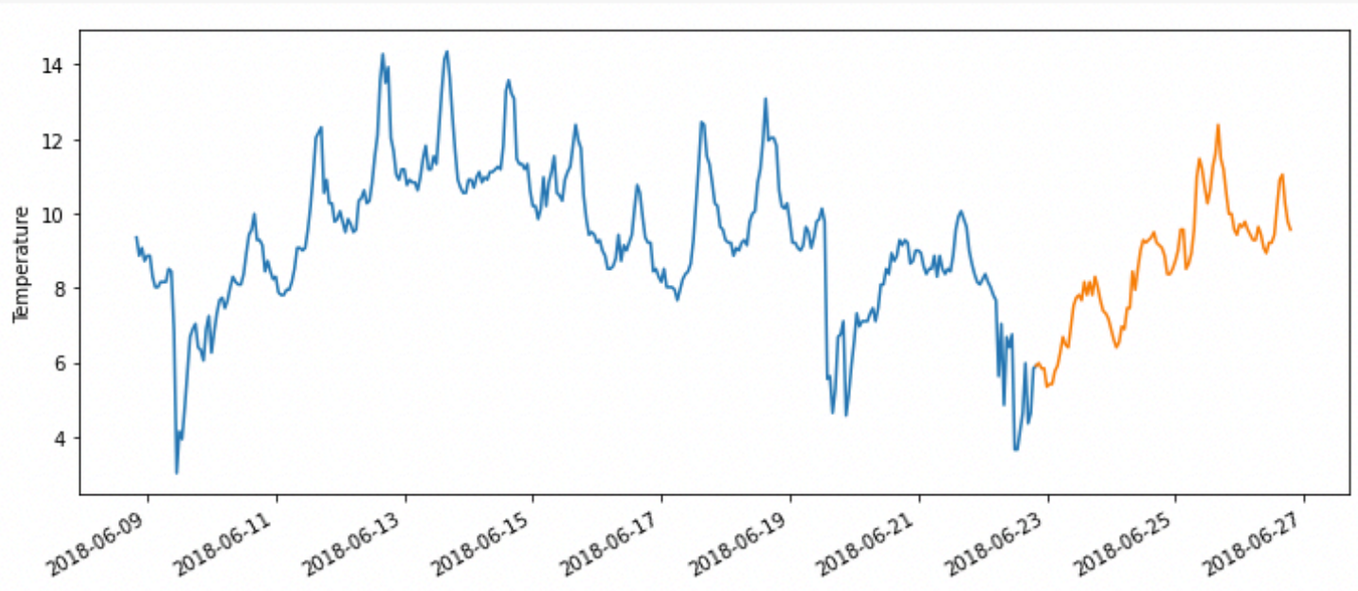
Field	date	HUFL	HULL	MUFL	MULL	LUFL	LULL	OT
Description	The recorded **date**	High UseFul Load	High UseLess Load	Middle UseFul Load	Middle UseLess Load	Low UseFul Load	Low UseLess Load	Oil Temperature (target)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17420 entries, 0 to 17419
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   date    17420 non-null  object 
 1   HUFL    17420 non-null  float64
 2   HULL    17420 non-null  float64
 3   MUFL    17420 non-null  float64
 4   MULL    17420 non-null  float64
 5   LUFL    17420 non-null  float64
 6   LULL    17420 non-null  float64
 7   OT      17420 non-null  float64
dtypes: float64(7), object(1)
memory usage: 1.1+ MB
```

```python
import matplotlib.pyplot as plt

seq_len = 336
pred_len = 96

plt.figure(figsize=(12,5))
plt.ylabel('Temperature')
data_to_show = etth1_df.set_index(pd.to_datetime(etth1_df['date']))
plt.plot(data_to_show['OT'][-(seq_len + pred_len):-(pred_len - 1)])
plt.plot(data_to_show['OT'][-pred_len:])
plt.gcf().autofmt_xdate()
plt.show()
```



3. 비교 분석: Informer

- 저자가 제공한 Pytorch 구현체를 사용.
- 하이퍼 파라미터를 일부 튜닝해봤지만, 논문에서 제시된 수치 이상으로 좋은 성능을 보이지는 않았다.

```
train 8209
val 2785
test 2785
     iters: 100, epoch: 1 | loss: 0.4836132
     speed: 0.2774s/iter; left time: 682.8012s
     iters: 200, epoch: 1 | loss: 0.4369209
     speed: 0.1929s/iter; left time: 455.4192s
Epoch: 1 cost time: 58.21726870536804
Epoch: 1, Steps: 256 | Train Loss: 0.5256339
Vali Loss: 1.3349595 Test Loss: 1.2772347
Validation loss decreased (inf --> 1.334960).
Saving model ...
Updating learning rate to 0.0001
     iters: 100, epoch: 2 | loss: 0.4046569
     speed: 0.4413s/iter; left time: 973.0052s
     iters: 200, epoch: 2 | loss: 0.3291671
     speed: 0.1996s/iter; left time: 420.1458s
Epoch: 2 cost time: 50.99851846694946
Epoch: 2, Steps: 256 | Train Loss: 0.3850533
Vali Loss: 1.1381409 Test Loss: 1.3329719
Validation loss decreased (1.334960 -->
1.138141).  Saving model ...
Updating learning rate to 5e-05
     iters: 100, epoch: 3 | loss: 0.2727984
     speed: 0.4526s/iter; left time: 882.1787s
     iters: 200, epoch: 3 | loss: 0.2480427
     speed: 0.2028s/iter; left time: 374.9315s
Epoch: 3 cost time: 51.984522342681885
Epoch: 3, Steps: 256 | Train Loss: 0.2583291
Vali Loss: 1.1441238 Test Loss: 1.6071874
EarlyStopping counter: 1 out of 3
Updating learning rate to 2.5e-05
     iters: 100, epoch: 4 | loss: 0.2151522
     speed: 0.4673s/iter; left time: 791.1177s
     iters: 200, epoch: 4 | loss: 0.2171048
     speed: 0.2069s/iter; left time: 329.6631s
Epoch: 4 cost time: 53.344425201416016
Epoch: 4, Steps: 256 | Train Loss: 0.2248309
Vali Loss: 1.1967032 Test Loss: 1.7237124
EarlyStopping counter: 2 out of 3
Updating learning rate to 1.25e-05
     iters: 100, epoch: 5 | loss: 0.2164504
     speed: 0.4635s/iter; left time: 665.9992s
     iters: 200, epoch: 5 | loss: 0.2062252
     speed: 0.2081s/iter; left time: 278.1709s
Epoch: 5 cost time: 53.21356129646301
Epoch: 5, Steps: 256 | Train Loss: 0.2121836
Vali Loss: 1.2108760 Test Loss: 1.7752172
EarlyStopping counter: 3 out of 3
Early stopping
```

3. 비교 분석: Informer

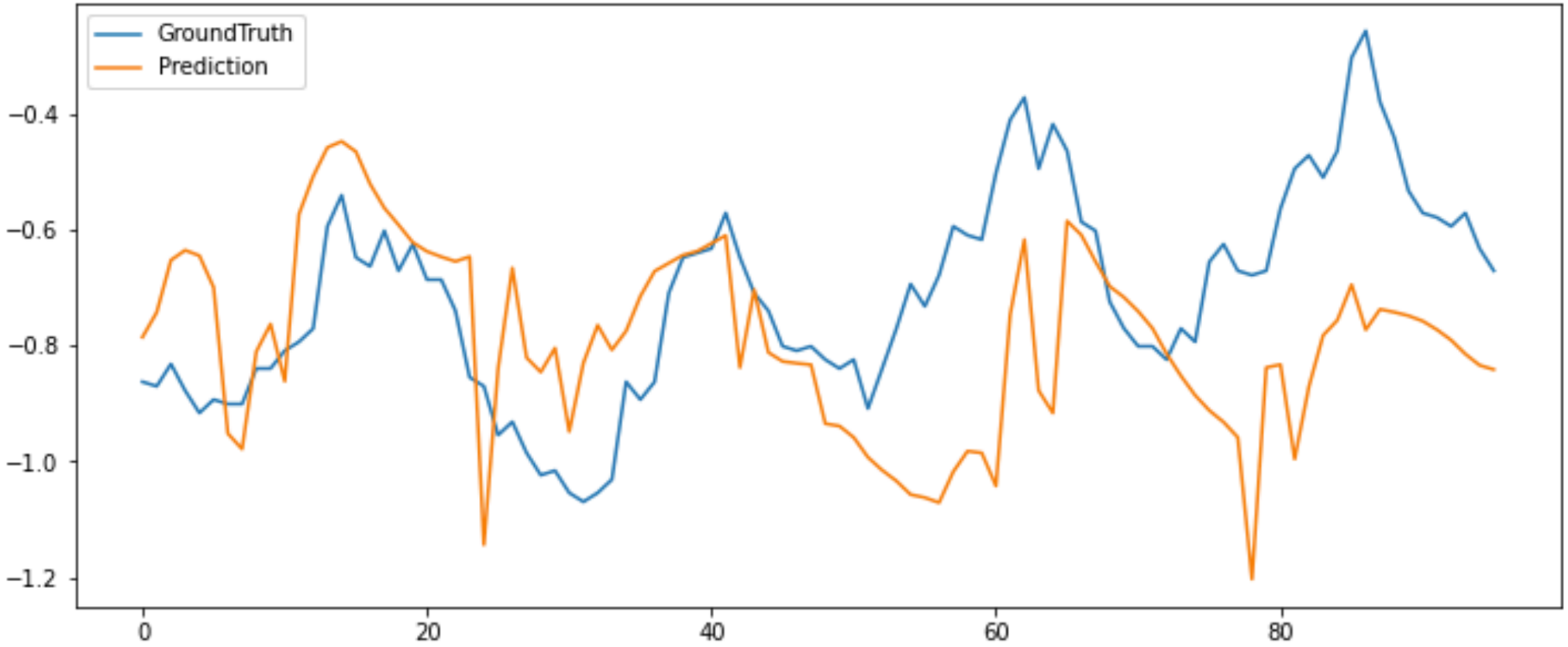
- MSE: 1.332184
- MAE: 0.974722
- 학습 시간: 337.404초(약 5.6분) 소요.
- 테스트 시간: 7.114초 소요.

```
test 2785
test shape: (87, 32, 96, 7) (87, 32, 96, 7)
test shape: (2784, 96, 7) (2784, 96, 7)
mse:1.3321846723556519, mae:0.9747226238250732
337.404188205
7.114572958999986
```

3. 비교 분석: Informer

```
pred = np.load('results/{}/pred.npy'.format(setting))
trues = np.load('results/{}/true.npy'.format(setting))

plt.figure(figsize=(12, 5))
plt.plot(trues[0,:,-1], label="GroundTruth")
plt.plot(pred[0,:,-1], label="Prediction")
plt.legend()
plt.show()
```



3. 비교 분석: DLinear

- 저자가 제공한 Pytorch 구현체를 사용.
- Informer 구현체와 달리 Ground Truth 데이터를 제공하지 않아서 구현체를 포크하여 코드베이스를 수정했다.

```
train 8209
val 2785
test 2785
      iters: 100, epoch: 1 | loss: 0.3700325
      speed: 0.0156s/iter; left time: 38.4609s
      iters: 200, epoch: 1 | loss: 0.3430304
      speed: 0.0058s/iter; left time: 13.6939s
Epoch: 1 cost time: 2.589700222015381
Epoch: 1, Steps: 256 | Train Loss: 0.4088630 Vali Loss:
0.6664551 Test Loss: 0.3947056
Validation loss decreased (inf ⟶ 0.666455).  Saving model
...
Updating learning rate to 0.0005
      iters: 100, epoch: 2 | loss: 0.4436035
      speed: 0.0453s/iter; left time: 99.7893s
      iters: 200, epoch: 2 | loss: 0.3813843
      speed: 0.0103s/iter; left time: 21.6703s
Epoch: 2 cost time: 3.420760154724121
Epoch: 2, Steps: 256 | Train Loss: 0.3486942 Vali Loss:
0.6676258 Test Loss: 0.3806984
EarlyStopping counter: 1 out of 3
Updating learning rate to 0.00025
      iters: 100, epoch: 3 | loss: 0.4300076
      speed: 0.0559s/iter; left time: 108.9302s
      iters: 200, epoch: 3 | loss: 0.3399374
      speed: 0.0131s/iter; left time: 24.2467s
Epoch: 3 cost time: 3.7878990173339844
Epoch: 3, Steps: 256 | Train Loss: 0.3413142 Vali Loss:
0.6808630 Test Loss: 0.3785386
EarlyStopping counter: 2 out of 3
Updating learning rate to 0.000125
      iters: 100, epoch: 4 | loss: 0.3456106
      speed: 0.0578s/iter; left time: 97.8918s
      iters: 200, epoch: 4 | loss: 0.4006236
      speed: 0.0063s/iter; left time: 10.0696s
Epoch: 4 cost time: 3.1247313022613525
Epoch: 4, Steps: 256 | Train Loss: 0.3382829 Vali Loss:
0.6576682 Test Loss: 0.3782265
Validation loss decreased (0.666455 ⟶ 0.657668).  Saving
model  ...
Updating learning rate to 6.25e-05
      iters: 100, epoch: 5 | loss: 0.3664558
      speed: 0.0327s/iter; left time: 46.9574s
      iters: 200, epoch: 5 | loss: 0.3501955
      speed: 0.0060s/iter; left time: 8.0670s
Epoch: 5 cost time: 2.008779764175415
Epoch: 5, Steps: 256 | Train Loss: 0.3371761 Vali Loss:
0.6641663 Test Loss: 0.3754797
EarlyStopping counter: 1 out of 3
Updating learning rate to 3.125e-05
      iters: 100, epoch: 6 | loss: 0.3483832
      speed: 0.0322s/iter; left time: 37.9818s
      iters: 200, epoch: 6 | loss: 0.3284534
      speed: 0.0057s/iter; left time: 6.1303s
Epoch: 6 cost time: 1.9635367393493652
Epoch: 6, Steps: 256 | Train Loss: 0.3364186 Vali Loss:
0.6579713 Test Loss: 0.3757418
EarlyStopping counter: 2 out of 3
Updating learning rate to 1.5625e-05
      iters: 100, epoch: 7 | loss: 0.3174990
      speed: 0.0330s/iter; left time: 30.5213s
      iters: 200, epoch: 7 | loss: 0.3271167
      speed: 0.0057s/iter; left time: 4.7395s
Epoch: 7 cost time: 1.9744904041290283
Epoch: 7, Steps: 256 | Train Loss: 0.3357472 Vali Loss:
0.6584820 Test Loss: 0.3754614
EarlyStopping counter: 3 out of 3
Early stopping
```

3. 비교 분석: DLinear

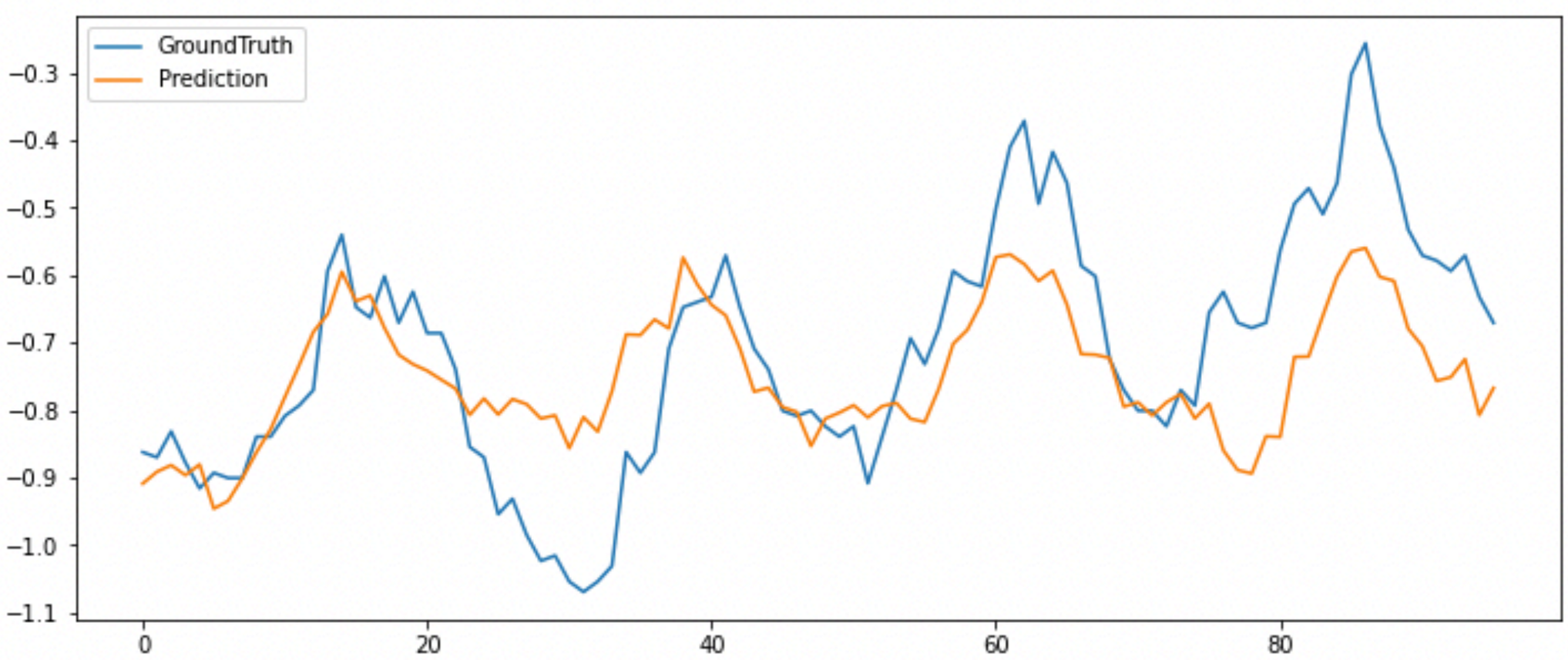
- MSE: 0.378226
- MAE: 0.400127
- 학습 시간: 34.949초 소요.
- 테스트 시간: 1.054초 소요.

```
test 2785
loading model
test shape: (87, 32, 96, 7) (87, 32, 96, 7)
test shape: (2784, 96, 7) (2784, 96, 7)
mse:0.3782263696193695, mae:0.40012705326080322
34.949470846999986
1.054309982999996
```

3. 비교 분석: DLinear

```
pred = np.load('./results/{}/pred.npy'.format(setting))
trues = np.load('./results/{}/true.npy'.format(setting))

plt.figure(figsize=(12, 5))
plt.plot(trues[0,:,-1], label="GroundTruth")
plt.plot(pred[0,:,-1], label="Prediction")
plt.legend()
plt.show()
```

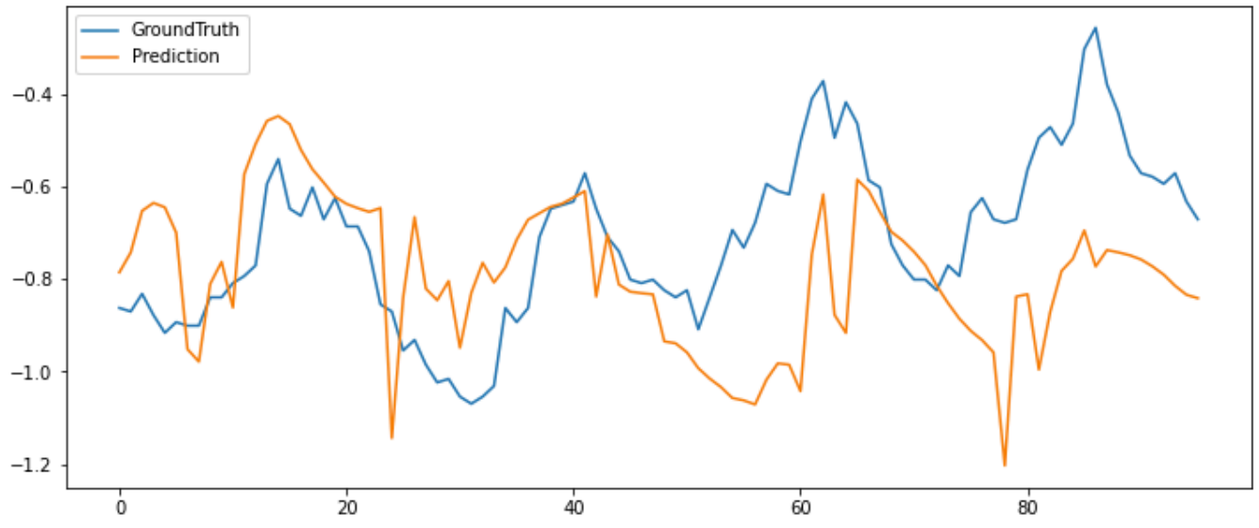


4. 결론

실제로 DLinear 모델이 Informer 모델보다 총체적으로 좋은 성능을 보였다.

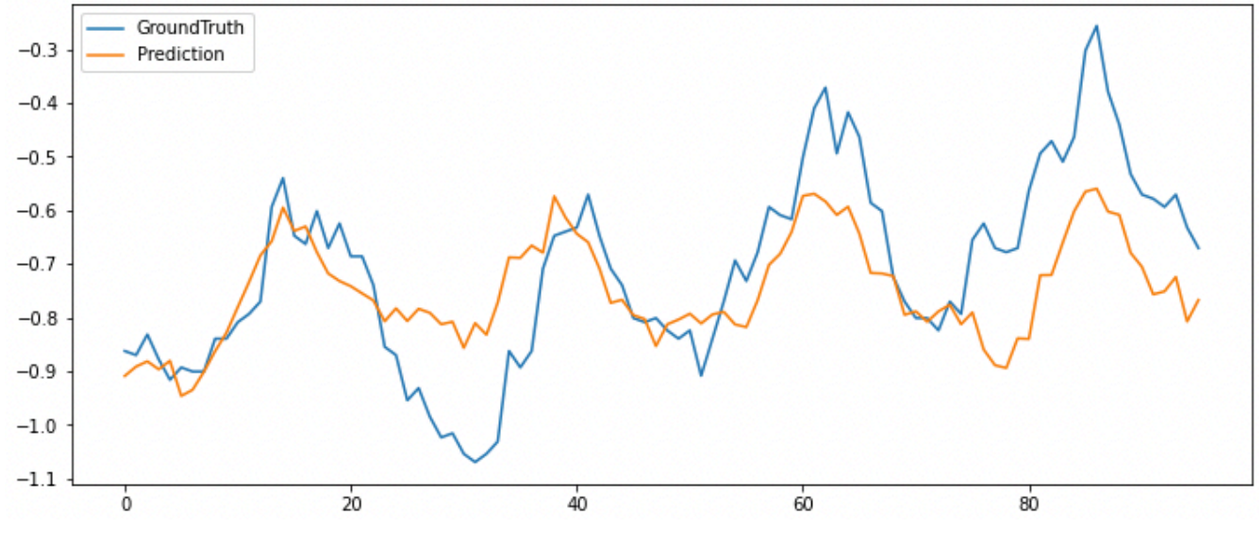
Informer

- MSE: 1.332184
- MAE: 0.974722
- 학습 시간: 337.404초(약 5.6분) 소요.
- 테스트 시간: 7.114초 소요.



DLinear

- MSE: 0.378226 **-71.6086%**
- MAE: 0.400127 **-58.9496%**
- 학습 시간: 34.949초 소요. **-89.6418%**
- 테스트 시간: 1.054초 소요. **-85.1841%**



4. 결론

- 시계열 예측에 선형 모델의 가능성을 보여준 연구.
- 다양한 데이터셋, 환경에서의 검증이 필요.
- 8월 17일: 논문 업데이트로 NLinear 모델 소개.
  LTSF-Linear 모델에 DLinear, NLinear 포함.
- 11월 23일: AAAI 2023 채택.
- 앞으로 주목해야 할 모델.

참고 문헌

- H. Zhou et al., “Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting”, 2021.

- A. Zeng et al., “Are Transformers Effective for Time Series Forecasting?”, 2022.

- A. Vaswani et al., “Attention Is All You Need”, 2017.