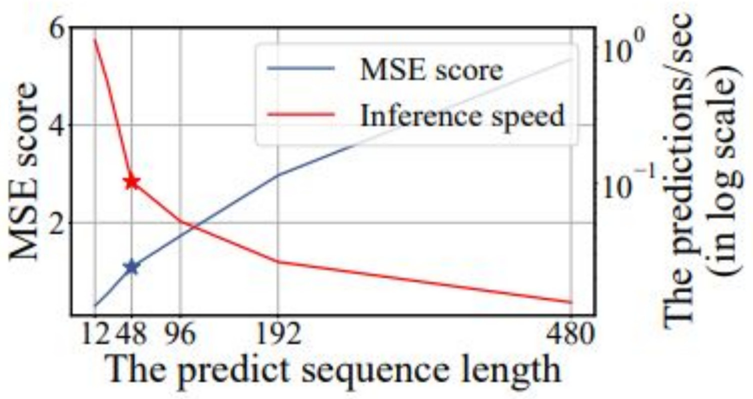
Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021, May).
In *Proceedings of the AAAI conference on artificial intelligence*
(Vol. 35, No. 12, pp. 11106-11115).

Background

Background

- Long sequence time-series forecasting (LSTF) is crucial across many domains
- How to enhance the prediction capacity of LSTF?
    - Long-range alignment ability
    - Efficient operations on long sequence inputs and outputs



Problems of Transformer

- The quadratic computation of self-attention $O(L^2)$

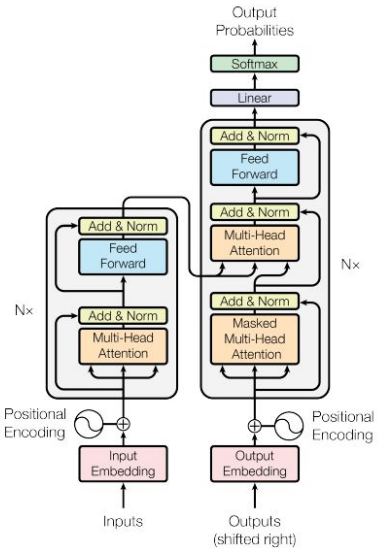
  → ProbSparse self-attention O(LlogL)

- The memory bottleneck in stacking layers for long inputs

  → Self-attention distilling operation

- The speed plunge in predicting long outputs

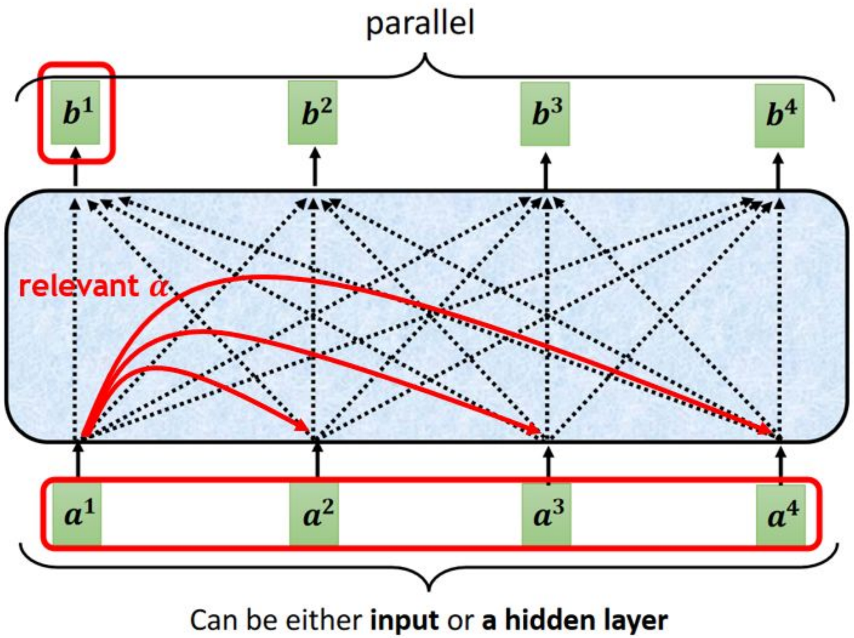
  → Generative-style decoder



Methodology

How does self-attention work?

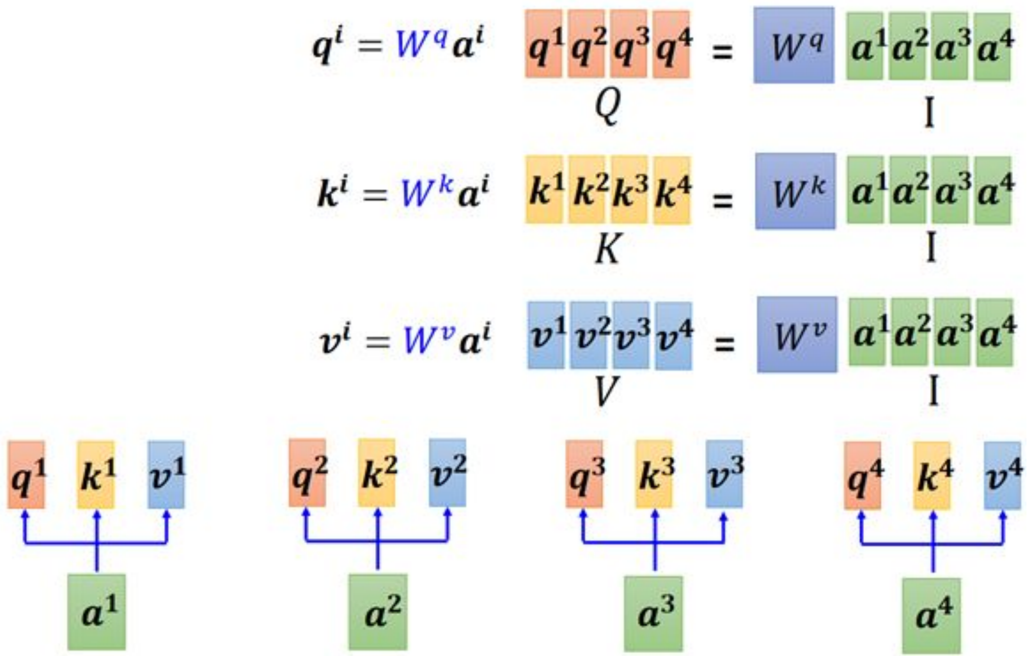
- Self-attention: Consider the context to get relevance.
  - [NIPS 2017] Attention is all you need!
- Give a sequence of data, return a sequence of corresponding attention score.



Ref: **李宏毅 ML2021 slides**

How does self-attention work?

- The canonical self-attention in is defined based on the tuple inputs.
    - query, key and value (That is, we have to learned the weight W^q, W^k, W^v)



Ref: 李宏毅 ML2021 slides

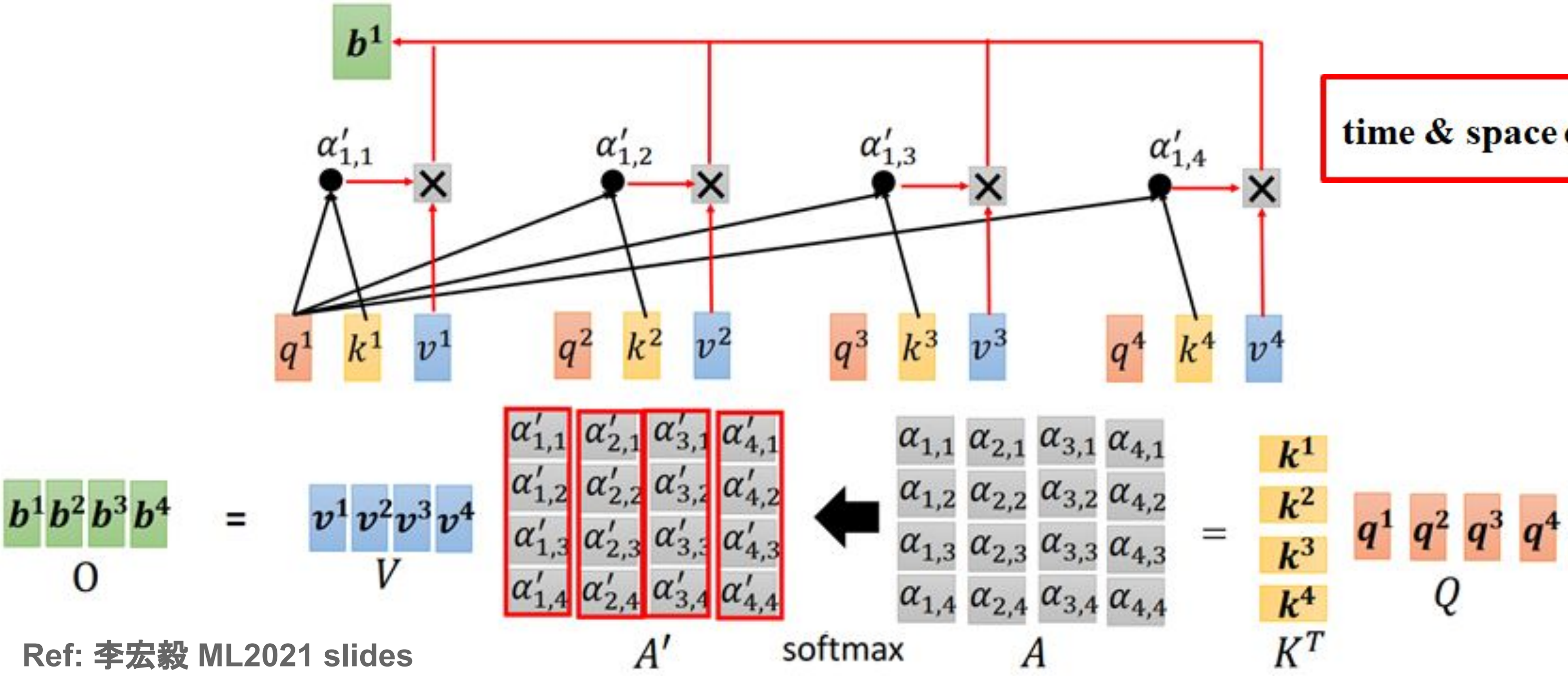
How does self-attention work?

- Performs the scaled dot-product as $A(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$

  Problem

  - where $Q \in \mathbb{R}^{L_Q \times d}$, $K \in \mathbb{R}^{L_K \times d}$, $V \in \mathbb{R}^{L_V \times d}$ and d is the input dimension

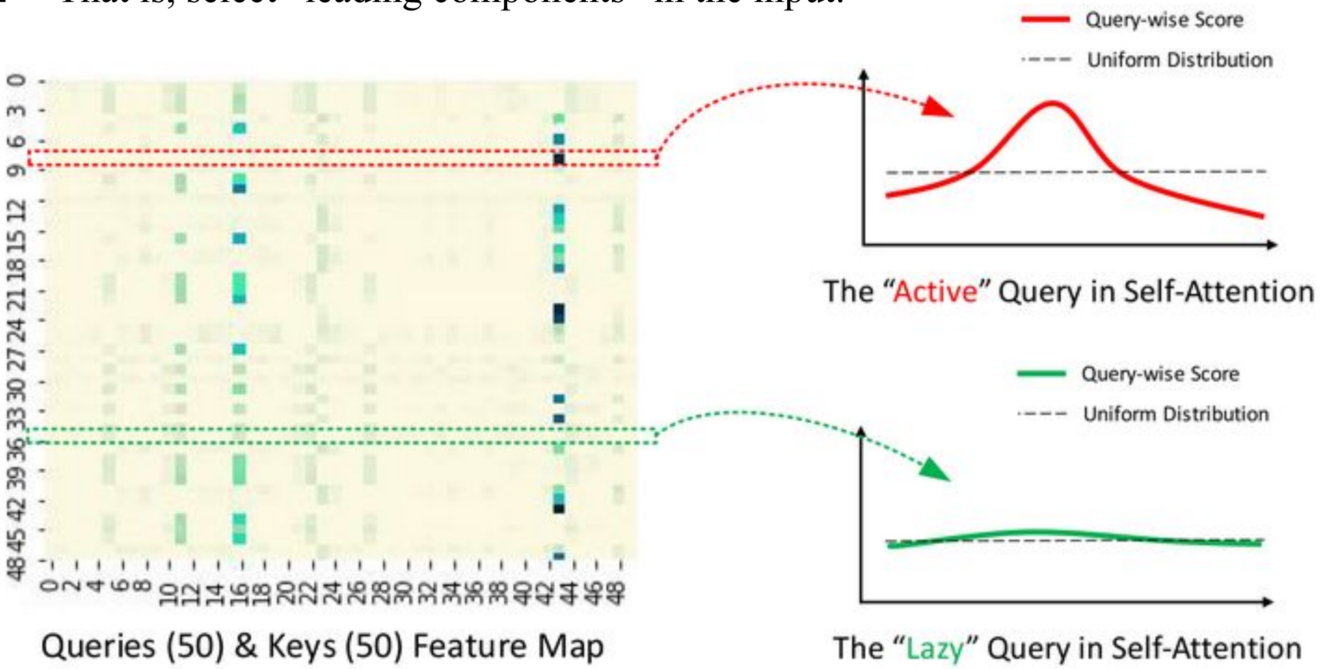
time & space complexity: $O(L^2)$



Ref: 李宏毅 ML2021 slides

Query Sparsity Measurement

- Self-attention probability has potential **sparsity** and also **long tail distribution**.
  - We want to find out “Active” Query.
    - That is, select “leading components” in the input.



Query Sparsity Measurement

- How to find out dominant dot product pairs
  - Use KL-divergence.

$\exp(\mathbf{q}_i \mathbf{k}_j^\top / \sqrt{d})$

$$\mathcal{A}(\mathbf{q}_i, \mathbf{K}, \mathbf{V}) = \sum_j \boxed{\frac{k(\mathbf{q}_i, \mathbf{k}_j)}{\sum_l k(\mathbf{q}_i, \mathbf{k}_l)}} \mathbf{v}_j = \mathbb{E}_{\boxed{p(\mathbf{k}_j | \mathbf{q}_i)}}[\mathbf{v}_j]$$

the i-th query's attention on all the keys

$$q(\mathbf{k}_j | \mathbf{q}_i) = 1/L_K$$

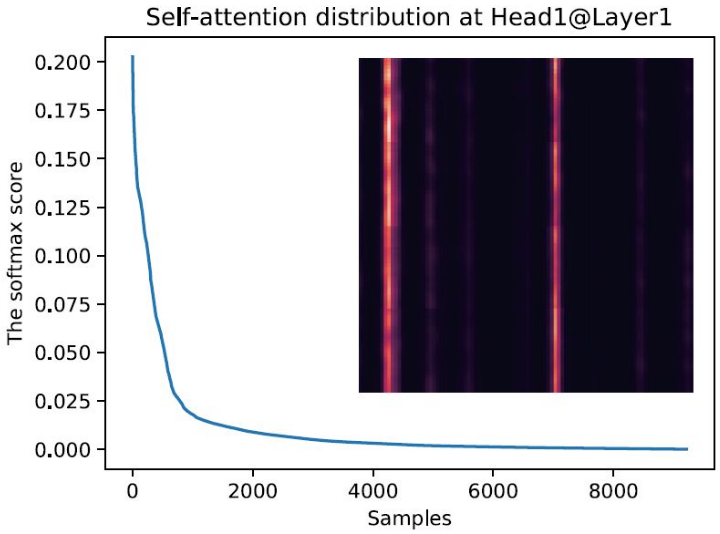
uniform distribution

$$KL(q||p) = \ln \sum_{l=1}^{L_K} e^{\mathbf{q}_i \mathbf{k}_l^\top / \sqrt{d}} - \frac{1}{L_K} \sum_{j=1}^{L_K} \mathbf{q}_i \mathbf{k}_j^\top / \sqrt{d} - \ln L_K$$

Get Top-u queries to calculate self-attention

How to reduce time & space complexity?

- Calcaute all dot-product pairs to find “active query”. $\longrightarrow$ $O(L^2)$ **NO ! !**
- Since long tail distribution, we only need to randomly sample $c \cdot \ln L$ keys
  - Active query is high-correlated with other keys.
    - Authors investigate vanilla Transformer on ETT dataset



How to reduce time & space complexity?

sample $c \cdot \ln L$ keys

Algorithm 1 ProbSparse self-attention

Require: Tensor $\mathbf{Q} \in \mathbb{R}^{m \times d}$, $\mathbf{K} \in \mathbb{R}^{n \times d}$, $\mathbf{V} \in \mathbb{R}^{n \times d}$

print set hyperparameter c, $u = c \ln m$ and $U = m \ln n$

randomly select U dot-product pairs from $\mathbf{K}$ as $\bar{\mathbf{K}}$

$O(L \ln L)$ set the $\boxed{\text{sample score } \bar{\mathbf{S}} = \mathbf{Q}\bar{\mathbf{K}}^\top}$

compute the measurement $M = \max(\bar{\mathbf{S}}) - \operatorname{mean}(\bar{\mathbf{S}})$ by row

set Top-u queries under M as $\bar{\mathbf{Q}}$

$O(L \ln L)$ set $\boxed{\mathbf{S}_1 = \operatorname{softmax}(\bar{\mathbf{Q}}\mathbf{K}^\top / \sqrt{d}) \cdot \mathbf{V}}$

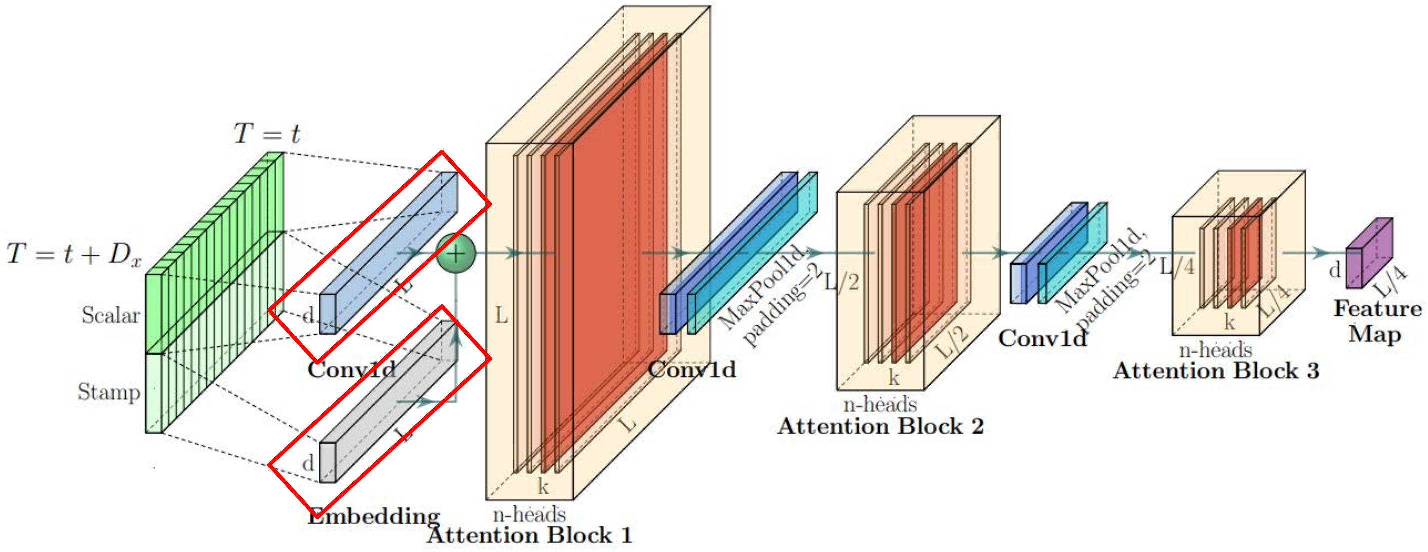
set $\mathbf{S}_0 = \operatorname{mean}(\mathbf{V})$

set $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_0\}$ by their original rows accordingly

Ensure: self-attention feature map $\mathbf{S}$.

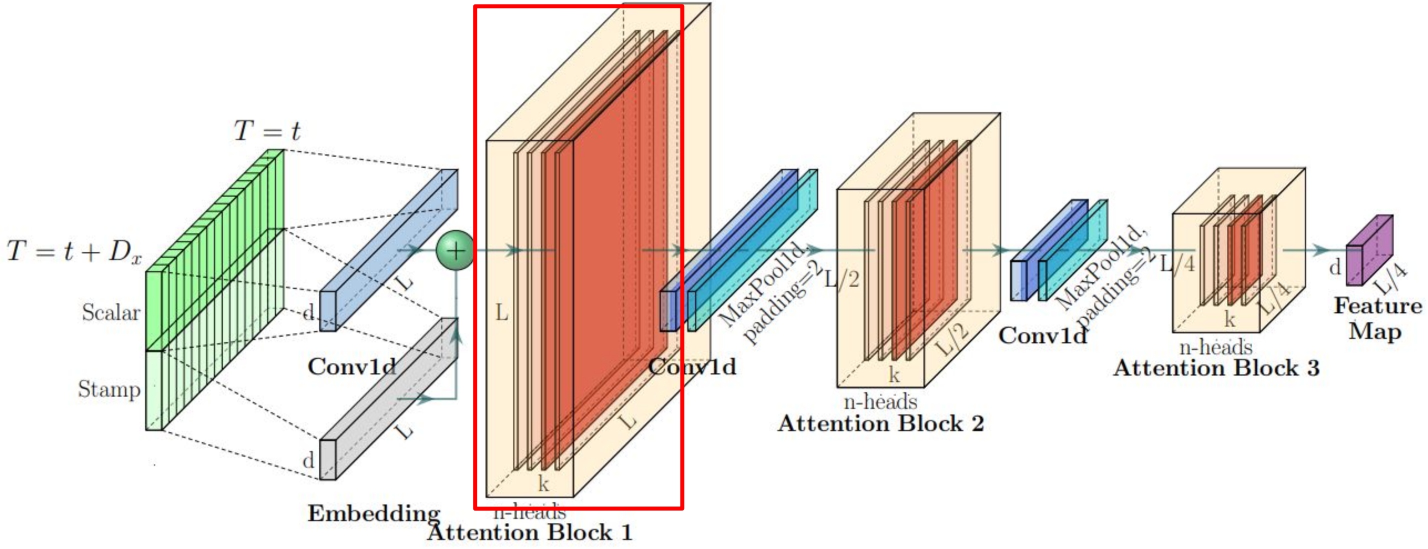
Encoder

- Input



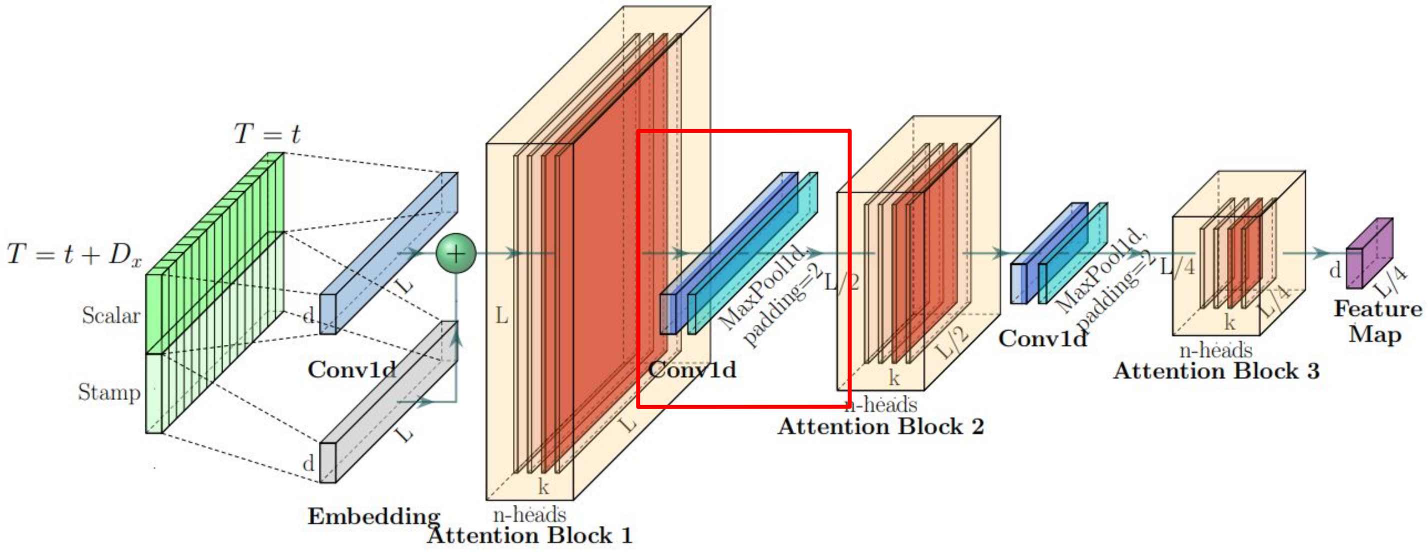
Encoder

- Probsparse self-attention



Encoder

- Distilling



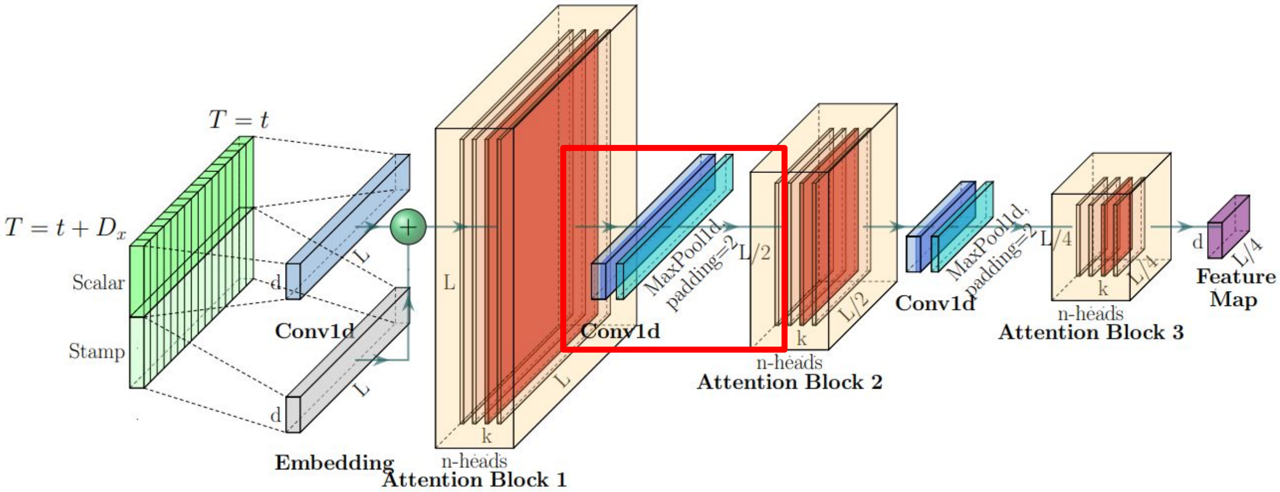
Self Attention Distilling

- Stack more **attention blocks** can get more detail features.
- Problem: the memory usage is $O(J \cdot L^2)$ if stack J layers
- Solution: Distilling operation

Self Attention Distilling

- Distilling operation to privilege the superior ones with dominating features

$$\mathbf{X}_{j+1}^{t} = \mathrm{MaxPool}\left(\mathrm{ELU}(\mathrm{Conv1d}([\mathbf{X}_{j}^{t}]_{\mathrm{AB}})) \right)$$



Self Attention Distilling

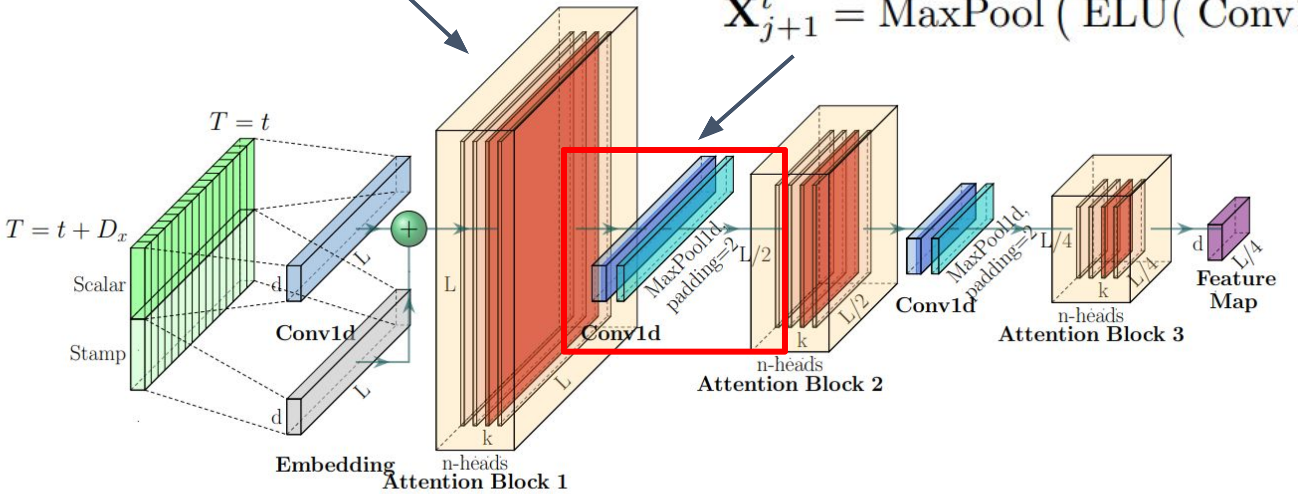
Active Query: $\mathbf{S}_1 = \mathrm{softmax}(\bar{\mathbf{Q}}\mathbf{K}^\top/\sqrt{d}) \cdot \mathbf{V}$

Lazy Query: $\mathbf{S}_0 = \mathrm{mean}(\mathbf{V})$

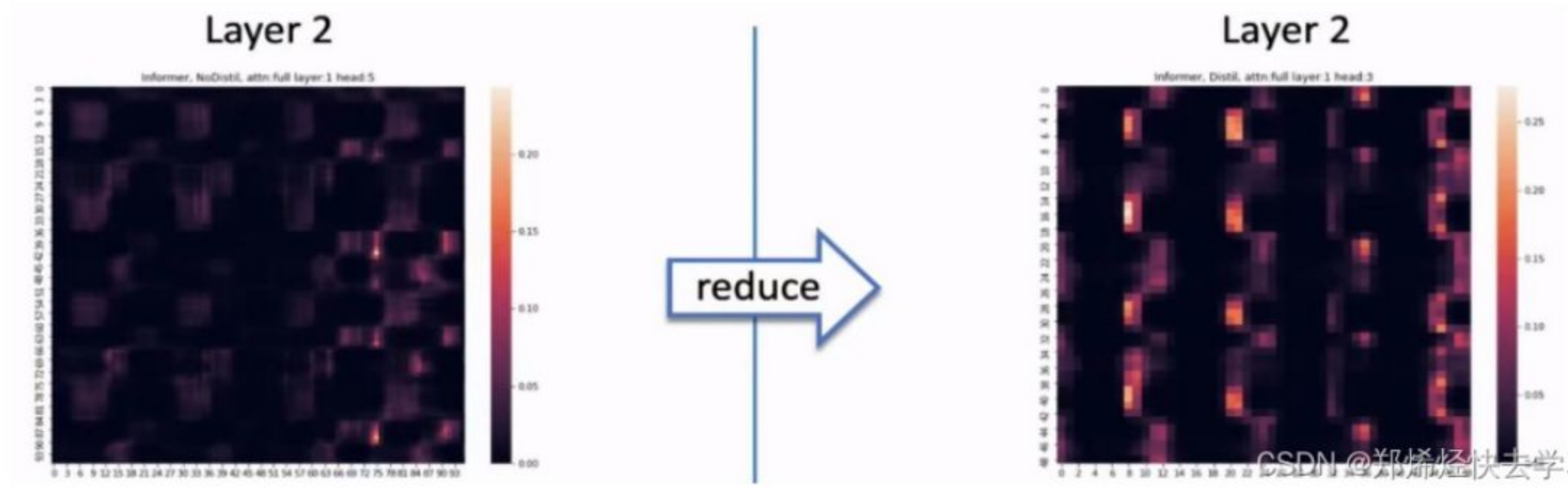
$\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_0\}$ by their original rows accordingly

- ❖ Lazy Queries are not important.
- ➔ Use maxpooling to make active queries dominate features map.
  - ➔ However, it lost some information.

$$\mathbf{X}_{j+1}^t = \mathrm{MaxPool}\left(\mathrm{ELU}(\mathrm{Conv1d}([\mathbf{X}_j^t]_{\mathrm{AB}})) \right)$$

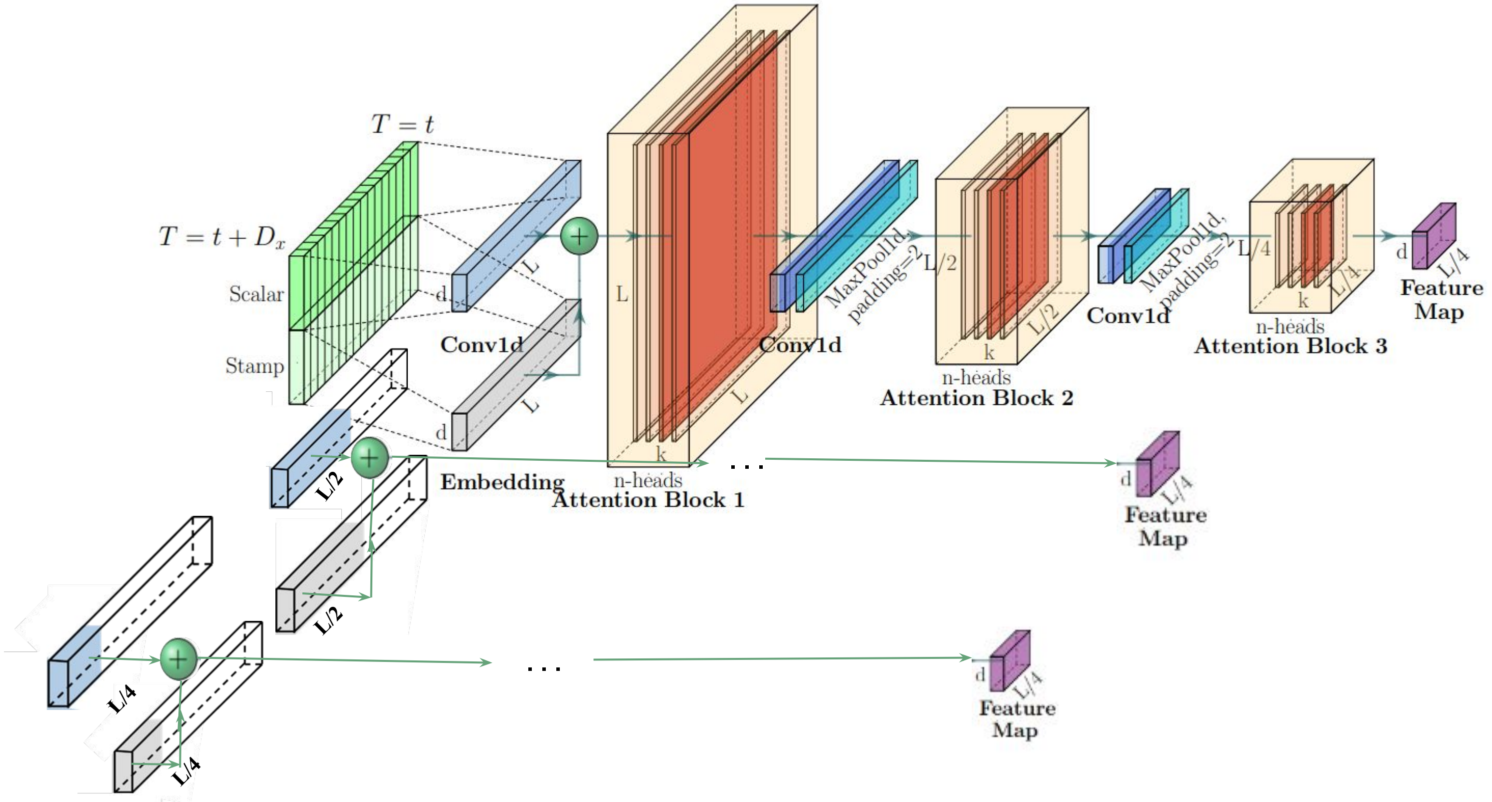


$$\mathbf{X}_{j+1}^{t} = \mathrm{MaxPool}\left(\mathrm{ELU}\left(\mathrm{Conv1d}([\mathbf{X}_{j}^{t}]_{\mathrm{AB}}) \right) \right)$$



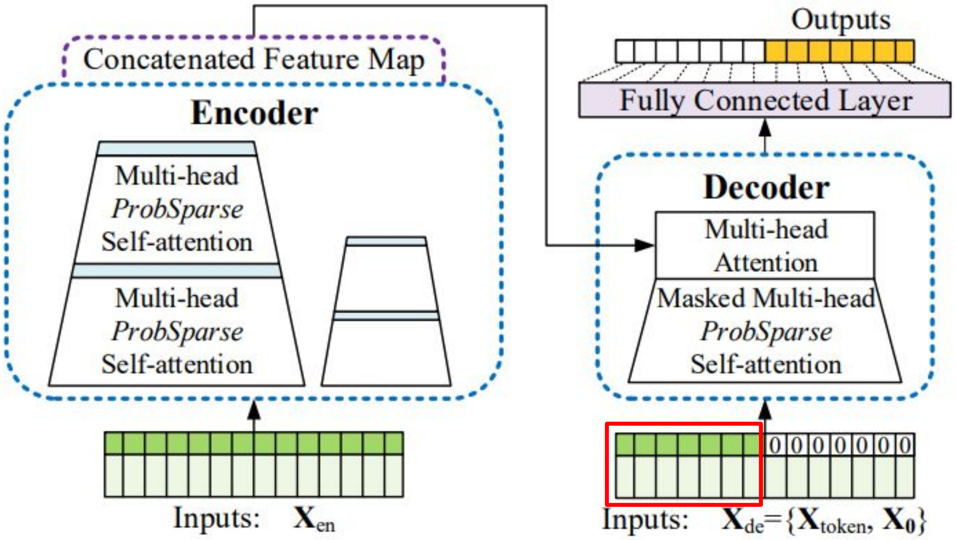
Self Attention Distilling

- Enhance the robustness of the distilling operation



- One Forward Procedure ➔ Use shorter input sequence(groundtruth) instead of specific flag as “start token” in decoder.

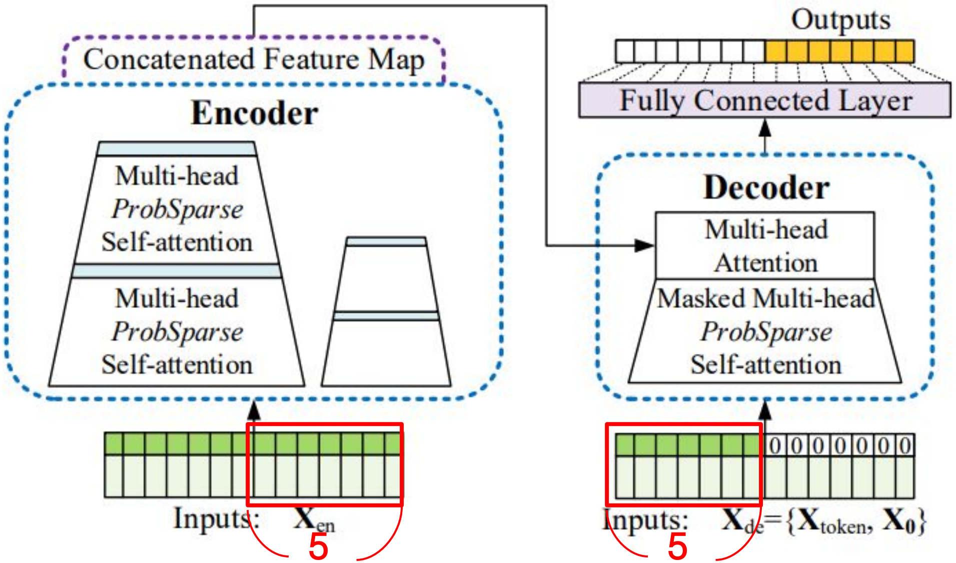
$$\mathbf{X}^t_{\mathrm{de}} = \mathrm{Concat}(\boxed{\mathbf{X}^t_{\mathrm{token}},}\mathbf{X}^t_{\mathbf{0}}) \in \mathbb{R}^{(L_{\mathrm{token}}+L_y)\times d_{\mathrm{model}}}$$



- One Forward Procedure

$$\mathbf{X}_{\text{de}}^t = \text{Concat}(\mathbf{X}_{\text{token}}^t, \mathbf{X}_{\mathbf{0}}^t) \in \mathbb{R}^{(L_{\text{token}}+L_y)\times d_{\text{model}}}$$

Predicit 7-day temperature, x_{tocken}^t = the 5 days before the forecasted time point.



Experiment

Experiment

1. Electricity Transformer Temperature(ETT)

2. Electricity Consuming Load (ECL)

3. Weather

Station	Date	Latitude	Longitude	SeaLevelPressure	WindDirection	WindSpeed	WetBulbTemperature
2907099999	1903-01-01T08:00:00	64.3333333	23.45	29.8	90	9	3
2907099999	1903-01-01T15:00:00	64.3333333	23.45	29.78	90	14	4
2907099999	1903-01-01T22:00:00	64.3333333	23.45	29.65	50	26	10
2907099999	1903-01-02T08:00:00	64.3333333	23.45	29.61	360	45	13
2907099999	1903-01-02T15:00:00	64.3333333	23.45	29.65	340	36	11

…

- All dataset will undergo **univariate** and **multivariate** time-series forecasting.

Experiment

- Univariate

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	

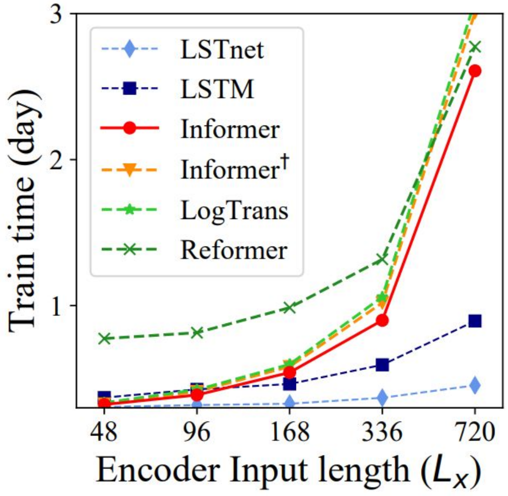
- Multivariate

Methods		Informer		Informer†		LogTrans		Reformer		LSTMa		LSTnet	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTh_1$	24	**0.577**	**0.549**	0.620	0.577	0.686	0.604	0.991	0.754	0.650	0.624	1.293	0.901
	48	**0.685**	**0.625**	0.692	0.671	0.766	0.757	1.313	0.906	0.702	0.675	1.456	0.960
	168	**0.931**	**0.752**	0.947	0.797	1.002	0.846	1.824	1.138	1.212	0.867	1.997	1.214
	336	1.128	0.873	**1.094**	**0.813**	1.362	0.952	2.117	1.280	1.424	0.994	2.655	1.369
	720	**1.215**	**0.896**	1.241	0.917	1.397	1.291	2.415	1.520	1.960	1.322	2.143	1.380
$ETTh_2$	24	**0.720**	**0.665**	0.753	0.727	0.828	0.750	1.531	1.613	1.143	0.813	2.742	1.457
	48	**1.457**	**1.001**	1.461	1.077	1.806	1.034	1.871	1.735	1.671	1.221	3.567	1.687
	168	3.489	**1.515**	3.485	1.612	4.070	1.681	4.660	1.846	4.117	1.674	**3.242**	2.513
	336	2.723	1.340	2.626	**1.285**	3.875	1.763	4.028	1.688	3.434	1.549	**2.544**	2.591
	720	**3.467**	**1.473**	3.548	1.495	3.913	1.552	5.381	2.015	3.963	1.788	4.625	3.709
$ETTm_1$	24	0.323	**0.369**	**0.306**	0.371	0.419	0.412	0.724	0.607	0.621	0.629	1.968	1.170
	48	0.494	0.503	**0.465**	**0.470**	0.507	0.583	1.098	0.777	1.392	0.939	1.999	1.215
	96	**0.678**	0.614	0.681	**0.612**	0.768	0.792	1.433	0.945	1.339	0.913	2.762	1.542
	288	**1.056**	**0.786**	1.162	0.879	1.462	1.320	1.820	1.094	1.740	1.124	1.257	2.076
	672	**1.192**	**0.926**	1.231	1.103	1.669	1.461	2.187	1.232	2.736	1.555	1.917	2.941
Weather	24	**0.335**	**0.381**	0.349	0.397	0.435	0.477	0.655	0.583	0.546	0.570	0.615	0.545
	48	0.395	0.459	**0.386**	**0.433**	0.426	0.495	0.729	0.666	0.829	0.677	0.660	0.589
	168	**0.608**	**0.567**	0.613	0.582	0.727	0.671	1.318	0.855	1.038	0.835	0.748	0.647
	336	**0.702**	**0.620**	0.707	0.634	0.754	0.670	1.930	1.167	1.657	1.059	0.782	0.683
	720	**0.831**	**0.731**	0.834	0.741	0.885	0.773	2.726	1.575	1.536	1.109	0.851	0.757
ECL	48	0.344	**0.393**	**0.334**	0.399	0.355	0.418	1.404	0.999	0.486	0.572	0.369	0.445
	168	0.368	0.424	**0.353**	**0.420**	0.368	0.432	1.515	1.069	0.574	0.602	0.394	0.476
	336	0.381	**0.431**	0.381	0.439	**0.373**	0.439	1.601	1.104	0.886	0.795	0.419	0.477
	720	0.406	0.443	**0.391**	**0.438**	0.409	0.454	2.009	1.170	1.676	1.095	0.556	0.565
	960	**0.460**	**0.548**	0.492	0.550	0.477	0.589	2.141	1.387	1.591	1.128	0.605	0.599
Count		33		14		1		0		0		2	

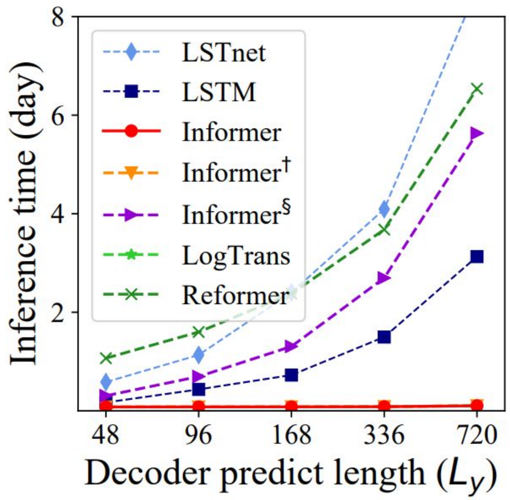
Experiment

- Time consumption

Training time:



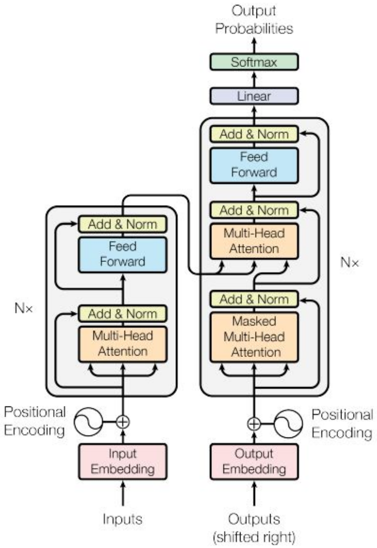
Inference time:

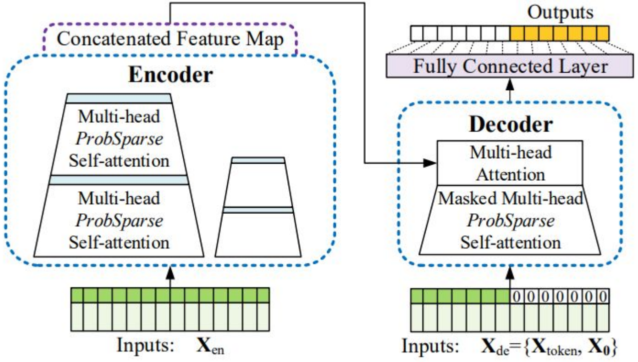


Conclusion

Conclusion

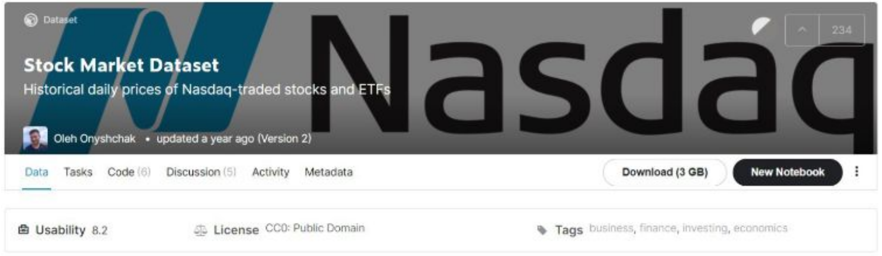
- Reduce time complexity → ProbSparse self-attention
- Reduce memory usage → ProbSparse self-attention
  & Self-attention distilling operation
- Reduce inference time → Generative style decoder

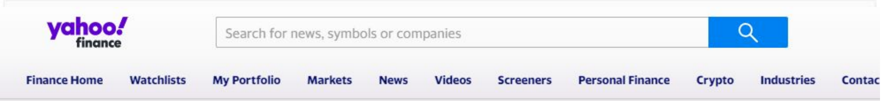




Final proposal

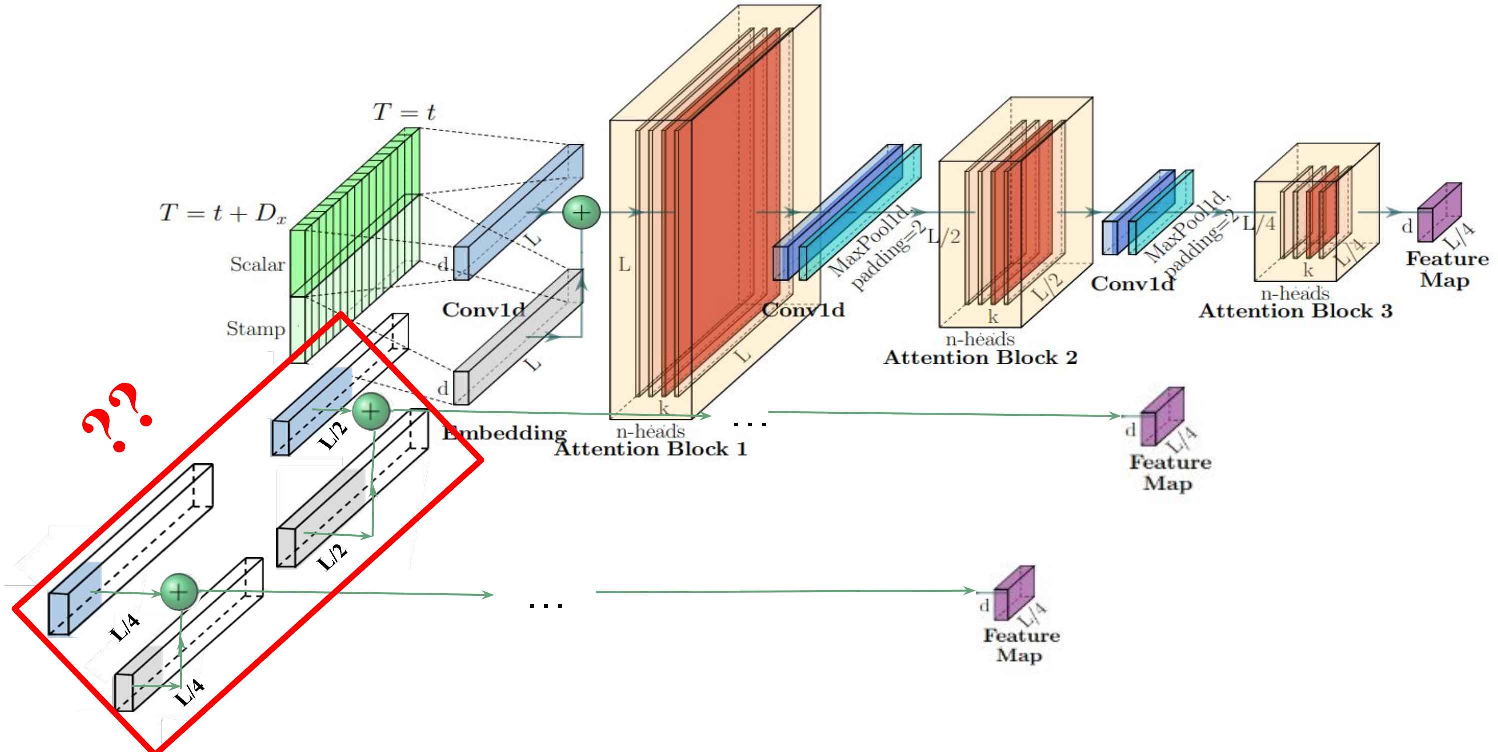
- Dataset: Stock
- We want to predict companies’ closing returns given stock features.
  - Date
  - Open
  - High
  - Low
  - Closing
  - Volume
- Univariate



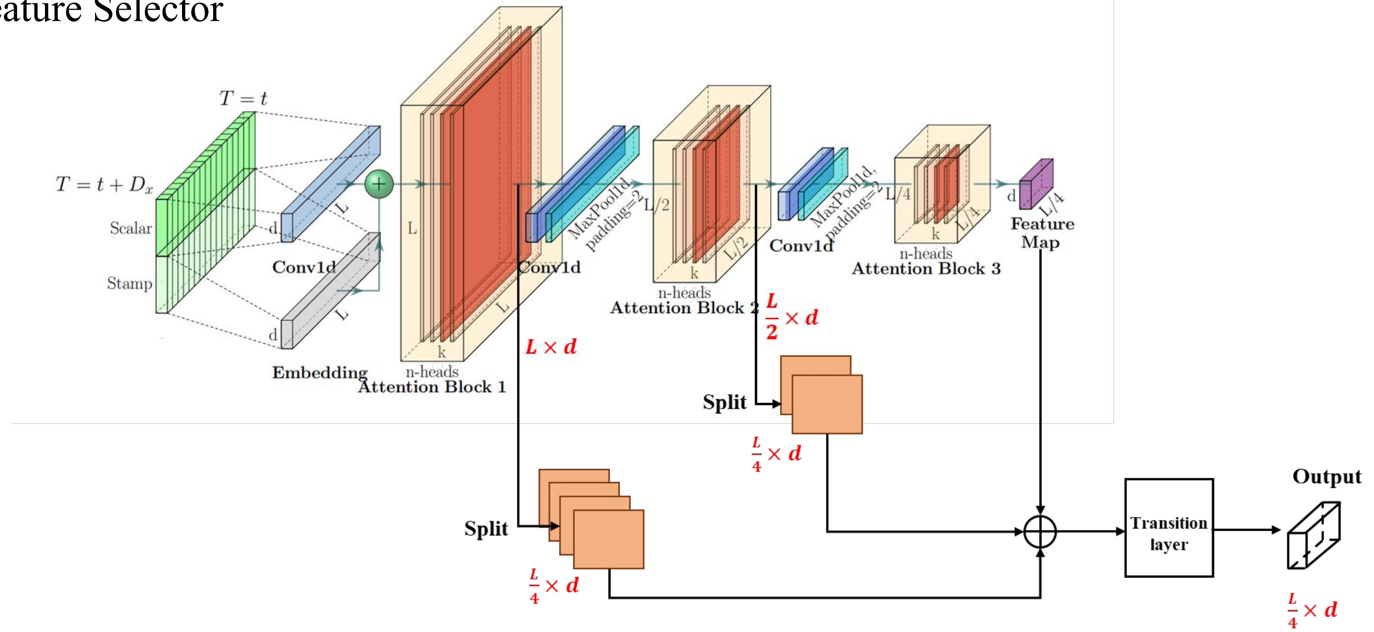


Date	Open	High	Low	Close	Adj Close	Volume
2020/2/6	30.99	31.56	29.56	30	30	2552630
2020/2/7	29.75	31.75	29.71	30.92	30.92	357500
2020/2/10	31.8	32	31	31.89	31.89	229510
2020/2/11	31.94	33.23	31.93	32.87	32.87	286300

- Problem: How to determine replicas of the main stack with halving inputs?
    - the first half, the second half or random selection?



- Feature combination of the low-layer and high-level features
  - High-layer features contain more semantic information.
  - Low-layer features contain more detail information.
  - ❖ [2019] Selective Feature Connection Mechanism: Concatenating Multi-layer CNN Features with a Feature Selector

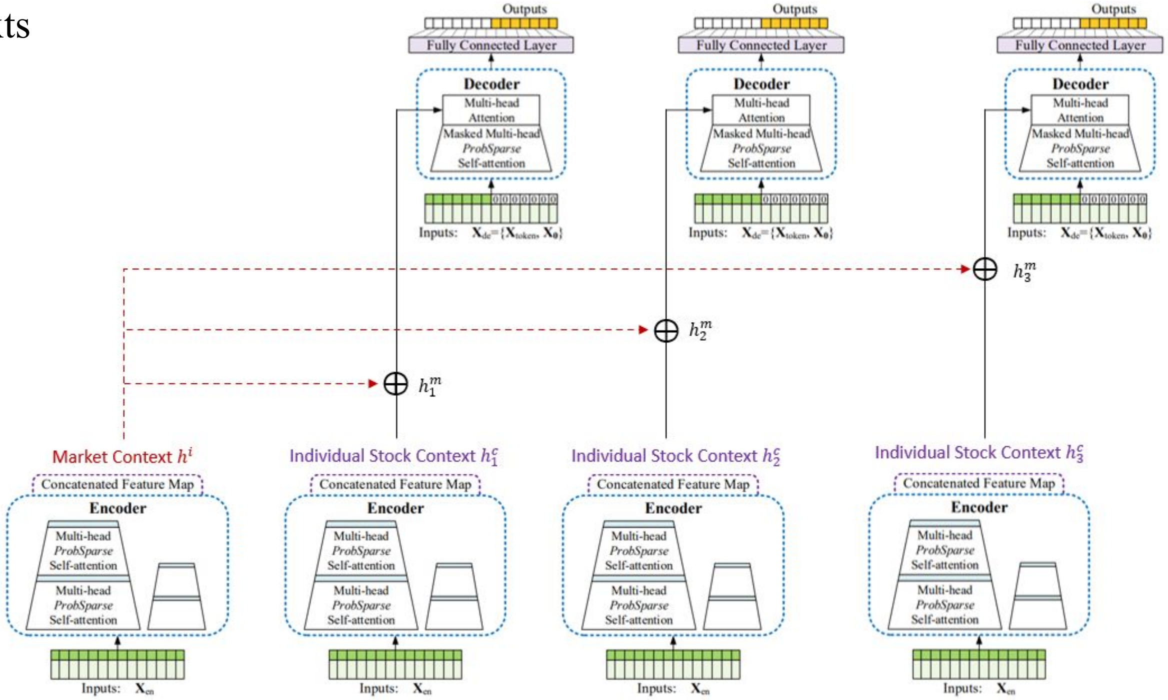


Proposal

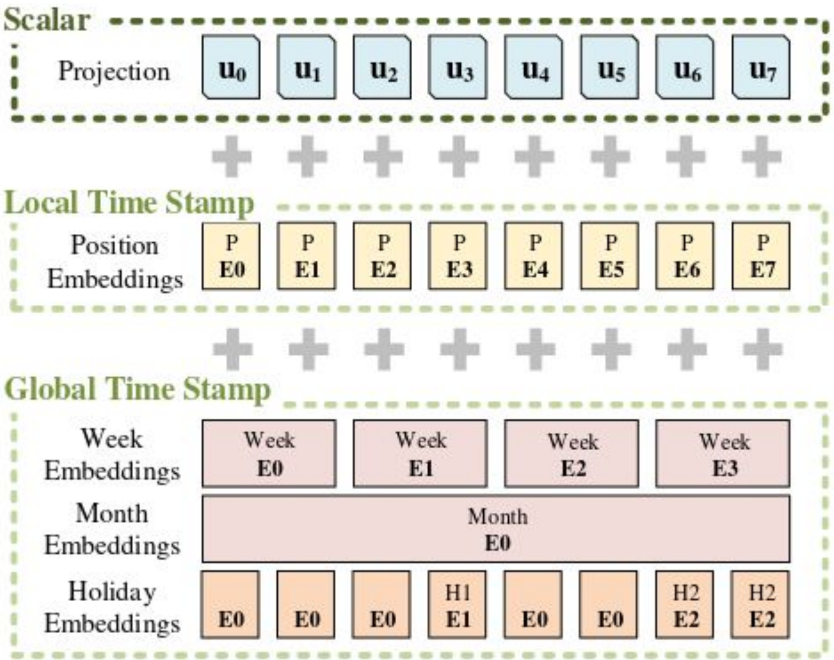
- How is stock dataset different from other dataset?
  - Stock prices are flutuated in every day.
  - Stocks in the same sector share a similar trend even though their prices are perturbed randomly in a short-term manner
    - Especially, stocks are highly correlated in a bull market (market index).
- How can we efficiently correlate multiple stocks for accurate stock movement prediction?

Proposal

- Expand input dimension: concatenate market index information as input
- Multi-Level Context Aggregation: $\mathbf{h}_u^m = \mathbf{h}_u^c + \beta \mathbf{h}^i$
  - [KDD'21] Accurate Multivariate Stock Movement Prediction via Data-Axis Transformer with Multi-Level Contexts



- Embedding method
  - Informer: **Local Position Embedding + Global Learnable Stamp Embedding**



Results

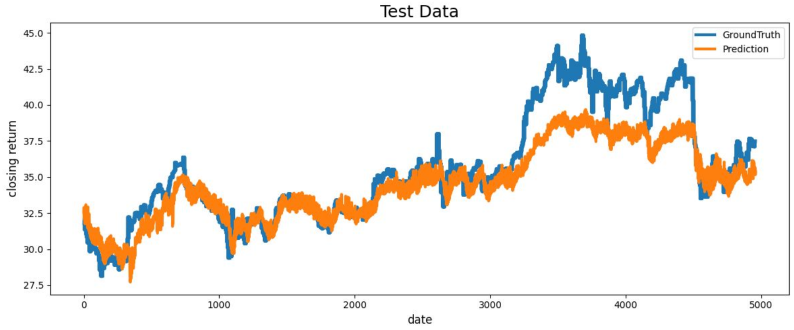
Proposal

- Hyperparameter
  - input sequence length of encoder: 64
  - start token length: 40
  - prediction sequence length: 5
  - num of encoder layers: 5,
  - num of decoder layers: 1,
  - training epoch: 20
  - loss function: MSE
  - learning rate: 1e-4 (decaying to 0.7x every epoch)
  - batch size: 32
  - early stopping patience: 5 (If validation data loss isn’t smaller than before five times, then stop.)
  - dropout: 0.05

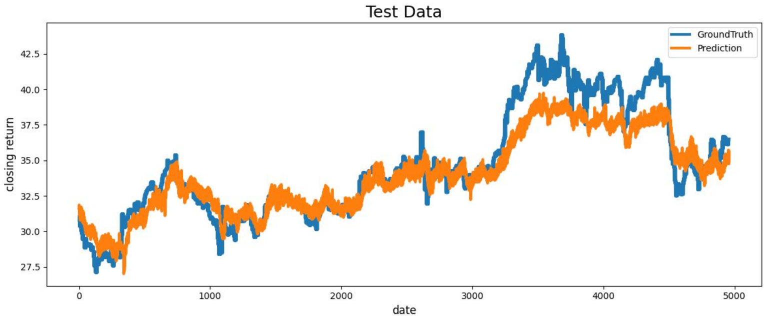
Results

- Predicted Pfizer's (PFE) stock using Johnson & Johnson's (JNJ), with a negative beta of -0.1128, indicating an inverse relationship due to industry competition.

Predict PFE with NASDAQ



Predict PFE with JNJ

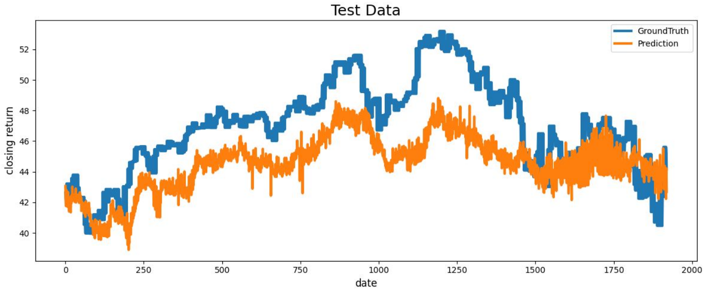


Results

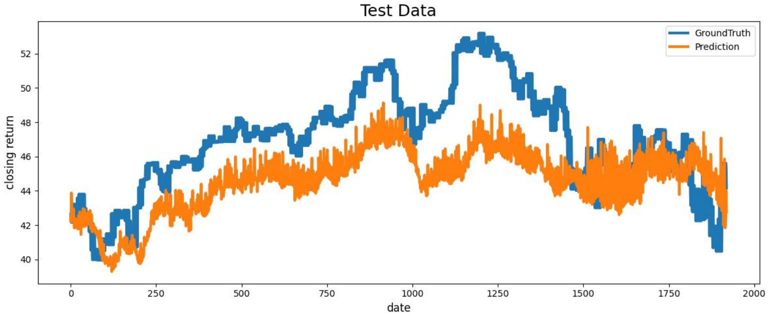
- Comparative analysis between the stocks and the overall market index (NASDAQ).

1. JPMorgan (JPM) for Finance

Predict JPM



Predict JPM with NASDAQ

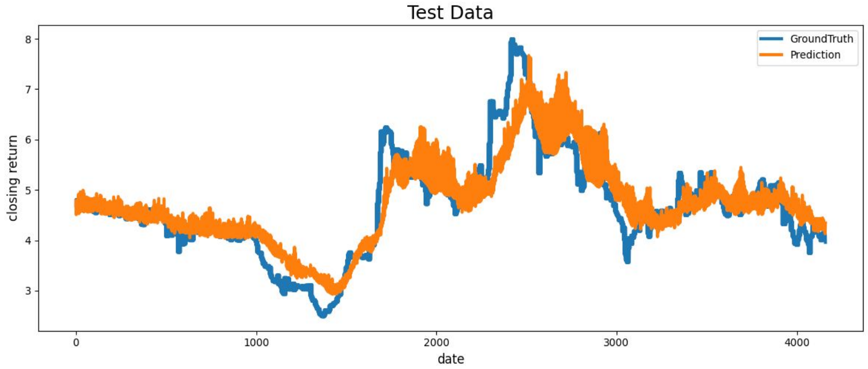


Results

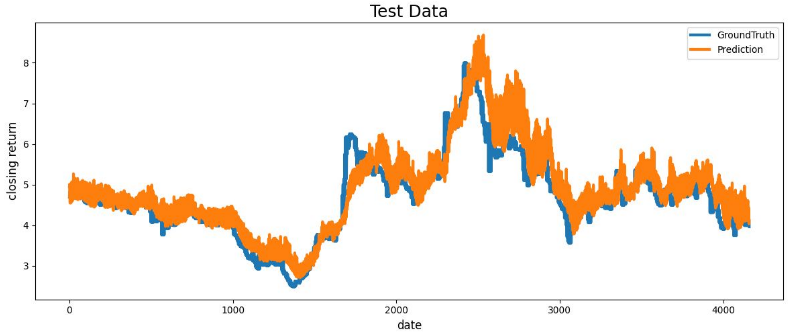
- Comparative analysis between the stocks and the overall market index (NASDAQ).

2. Drive Shack (DS) for Entertainment

Predict DS



Predict DS with NASDAQ



Results

- Comparative analysis between the stocks and the overall market index (NASDAQ).

3. ExxonMobil (XOM) for Energy

Predict XOM

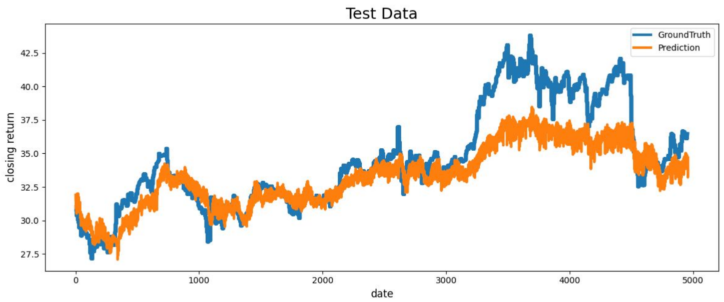
Predict XOM with NASDAQ

Results

- Comparative analysis between the stocks and the overall market index (NASDAQ).

4. Pfizer (PFE) for Pharmaceuticals

Predict PFE



Predict PFE with NASDAQ