国际人工智能会议
AAAI 2021 论文北京预讲会

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

Authors: Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang

Jianxin Li, Xiong Hui, Wancai Zhang

BEIHANG UNIVERSITY

Berkeley
UNIVERSITY OF CALIFORNIA

RUTGERS

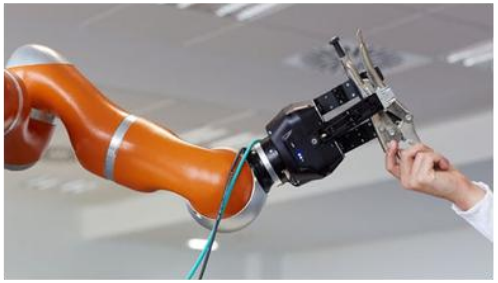
# CONTENT

- Background
- Motivation: why self-attention?
- Methods: the details of Informer
- Experiments
- Summary

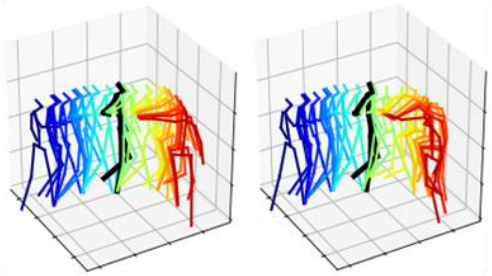
# What's the main topic of this paper?

- **Sequence prediction** is a problem that involves using historical sequence information to predict the next value or values in the sequence. It is a basic but important research problem.


Stock Market Prediction



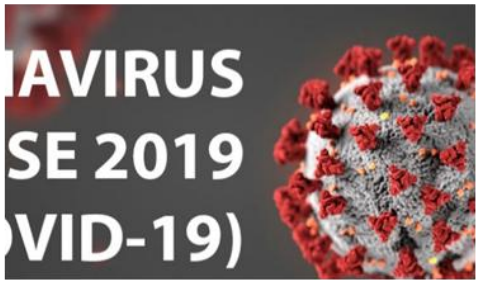
Robots Action Prediction



Human Position Prediction


Weather Prediction

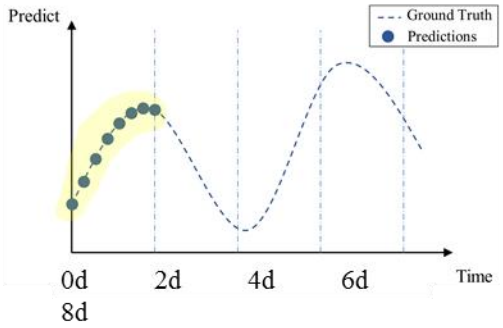
Supply Chain prediction




Society Event Prediction

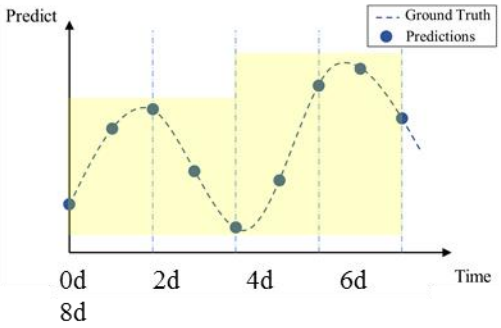
# An Example

**Electricity Transformer (over 100kv)** is a valuable asset for the energy company. The cost of a single unit can reach 30 million US dollars, which makes only conservative adjustments are allowed, i.e., one adjustment lasts 2 weeks, for stability.



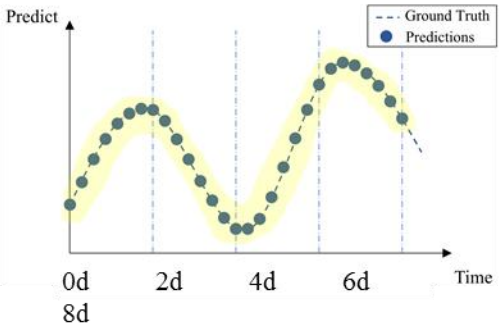

Near future predictions

(Limited adjustment)




Coarse predictions

(Inefficient adjustment)

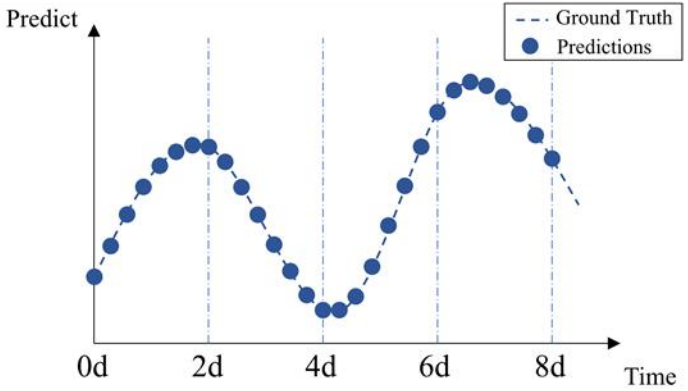


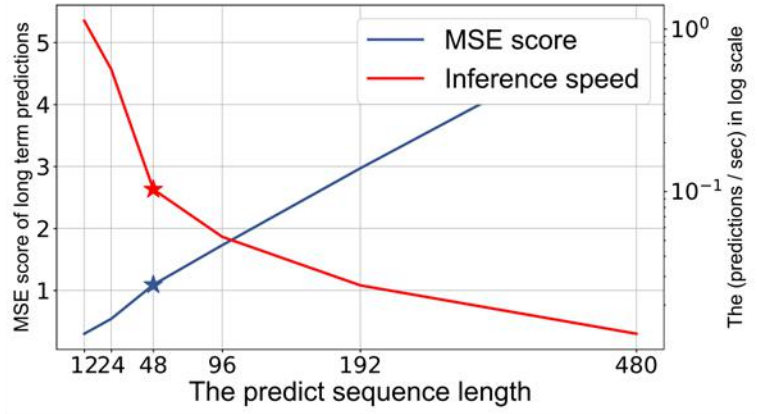

Long sequence predictions

(Proper adjustment)

# Long Sequence Predictions

**Problem: Long Sequence Time-series Forecasting (LSTF)** is an important problem that has not been well addressed in the time-series analysis field for a long time. On the other hand, because of the "huge data, high frequency, long prediction" features of the real-world time-series data, how to solve this problem is one of the key limitations in big data applications.



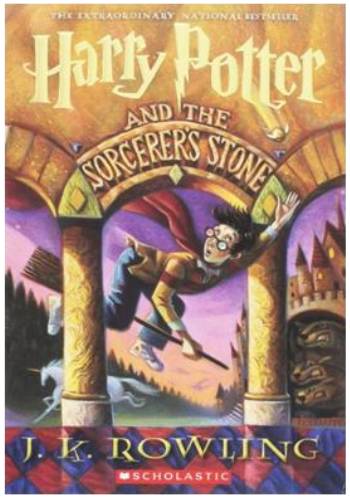




As the length of the data sequence increases, the inference speed of LSTM decreases rapidly, resulting in the inability to predict long sequences with limited computing power and time successfully. Meanwhile, the continuous accumulation of error causes the MSE score to increase rapidly, making the results unusable.

# Another Similar Problem …

- **Discrimination: Long Sequence Input Learning (LSIL)** is another important problem that remains to be solved in the machine learning field. It is not necessary related to the LSTF problem but the literature always refers to LSIL when talking about the long sequence.
- **Example:**

Input




76944 words

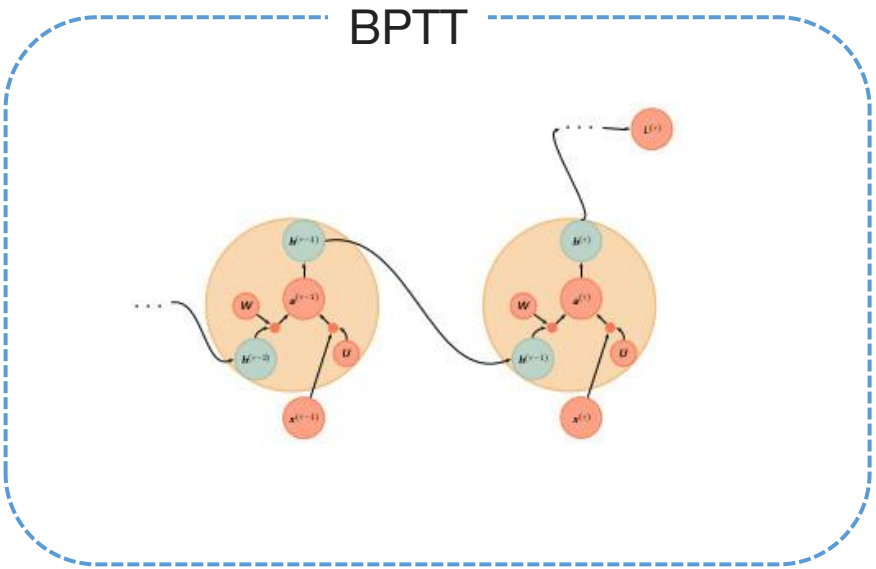
Output

- **Classification:** All-age Novels
- **Title Summary:** A boy wizard begins training and must battle for his life with the Dark Lord who murdered his parents.
- **Short Summary:** A boy who learns on his eleventh birthday that he is the orphaned son of two powerful wizards and possesses unique magical powers of his own. He is summoned from his life as an unwanted child to become a student at Hogwarts, an English boarding school for wizards. There, he meets several friends who become his closest allies and help him discover the truth about his parents' mysterious deaths.

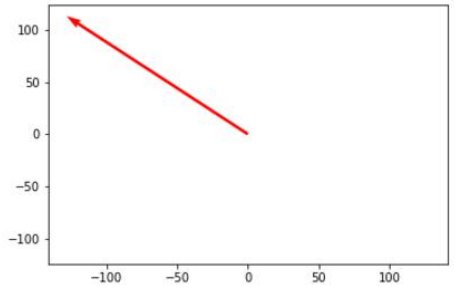
# The previous research on LSIL

## Literature Review of Long Sequence Input Learning Problem (LSIL)

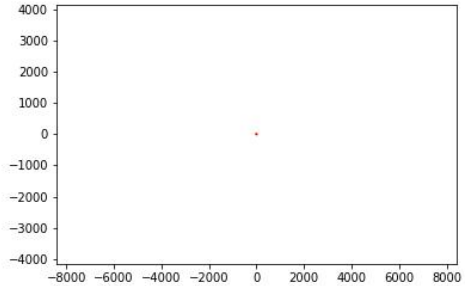
We capture the long term dependencies with gradient descent, however, is difficult because the gradients computed by BPTT tend to vanish or explode (Hochreiter et al., 2001).




It become worse if there are more than 128 BPTT steps.




Gradient Vanishing




Gradient Exploding

# The previous research on LSIL

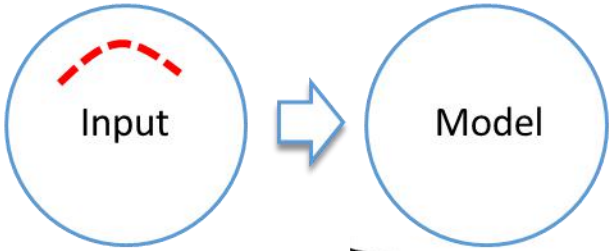
## Literature Review of Long Sequence Input Learning Problem (LSIL)

We capture the long term dependencies with gradient descent, however, is difficult because the gradients computed by BPTT tend to vanish or explode (Hochreiter et al., 2001).

- **Practice:** Truncate Sequences / Summarize Sequences / Random Sampling. These methods reduce the length of the input data to make it tractable.
  **Regression:** fast F2F '14 / additive F2F '18. These methods can perform the large-scale regression.

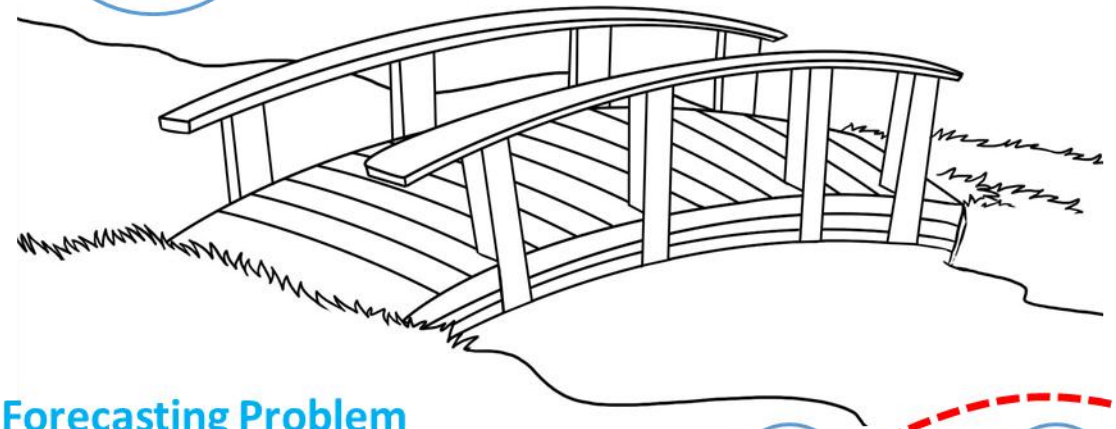
- **RNN-based model:** Periodically States '16 / Truncated BPTT '17 / Auxiliary Losses '18 / Recurrent Highway Networks '19 / Bootstrapping Regularizer '20. They try to only calculate selected hidden states or storage hidden states in the memory.
- **Attention-based model:** Naive Attention '14 / End-to-end Memory '15 / ED network '17. They improve the gradient flow by shortening the path that relevant information needs to traverse in the computation graph.
- **CNN-based model:** Wavenet '16 / TCNnet 18' / Seq-U-Net 19'. They use the convolutional neural networks with dilated filters to capture the long term dependencies. Their receptive fields grow exponentially with the number of layers.
- **Self-attention model**: Transformer '17 / Bert '19 / Transformer-xl '19. The calculation of self-attention is quadratic in the length of the sequence inputs.

# The research gap between LSTF and LSIL problem



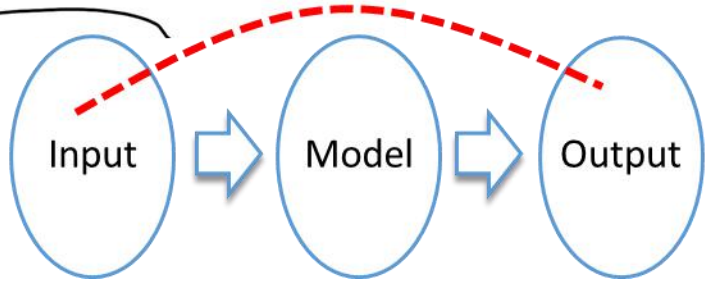

## Long Sequence Input Learning Problem

Enhance the model's ability to handle massive inputs. The most important thing is to capture the long term independencies among the long sequence inputs.



## Long Sequence Forecasting Problem

Enhance the model's prediction capacity to have a more extended output. The most important thing is to capture the long term independencies between the long sequence output and the inputs.



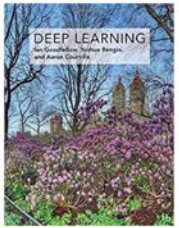
# CONTENT

- Background
- Motivation: why self-attention?
- Methods: the details of Informer
- Experiments
- Summary

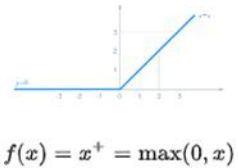
# How do human beings tackle the forecasting problem?

**Problem Setting:** Long Sequence Forecasting predicts the long sequence output from massive long sequence inputs.

**How do we answer a question?**

What is Relu()? → Machine Learning → 

→ 

→ 

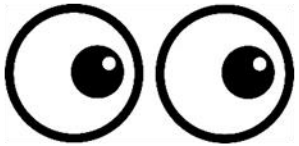

**How do we plan gifts for Christmas day?**

The Gifts list for this year? → Last year? → Wish list? → Popular things? → 

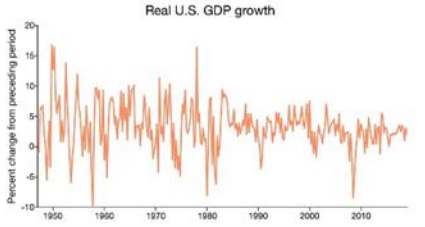

We will have some feelings of what you have seen when it comes to the prediction.

# The intuitive introduction of attention

- **Attention Mechanism** is arguably one of the most powerful concepts in the deep learning field nowadays. It is based on a common intuition that we "attend to" a certain part when processing a large amount of information.

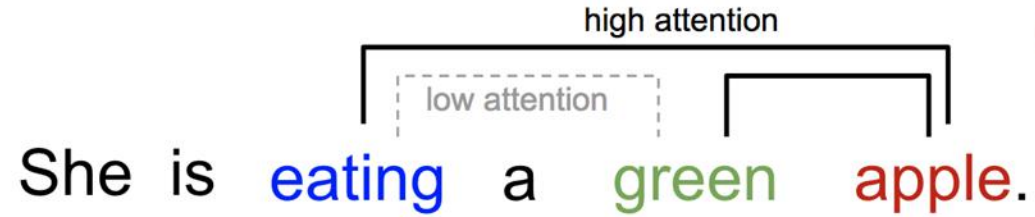


Attention




- **Example:**

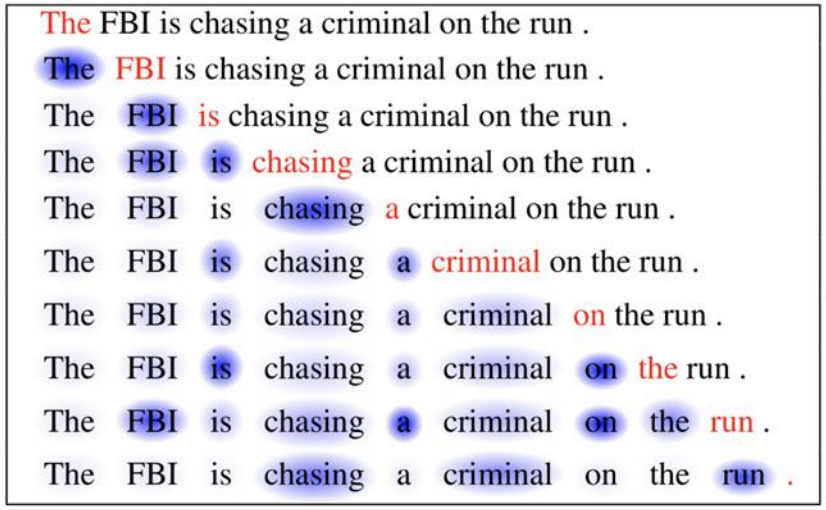
We can explain the relationship between words in one sentence or close context. When we see "eating", we expect to encounter a food word very soon. The color term describes the food, but probably not so much with "eating" directly.

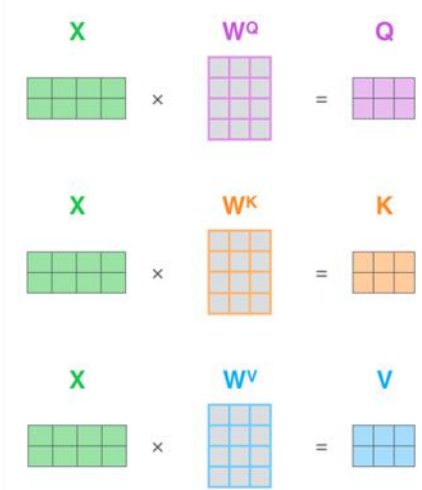


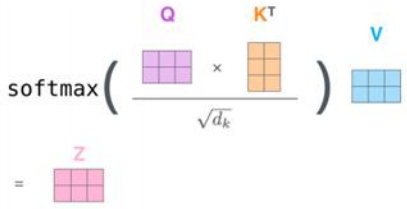

direct access!

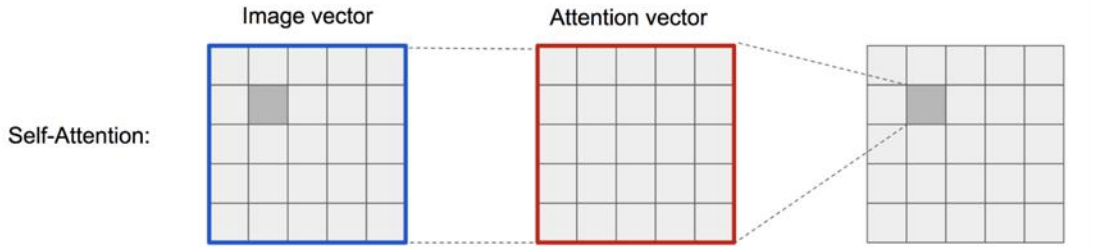
Acknowledgement from https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

# The self-attention in NLP/CV field



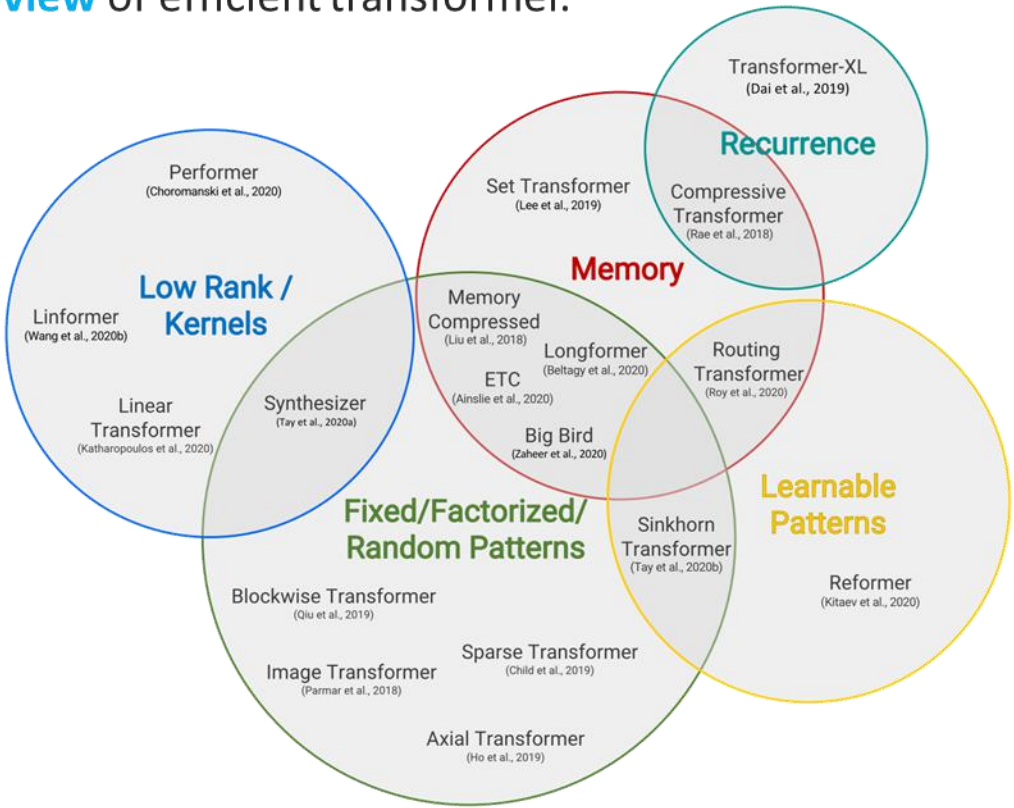




Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/

# The previous research on efficient self-attention

**Literature Review** of efficient transformer.

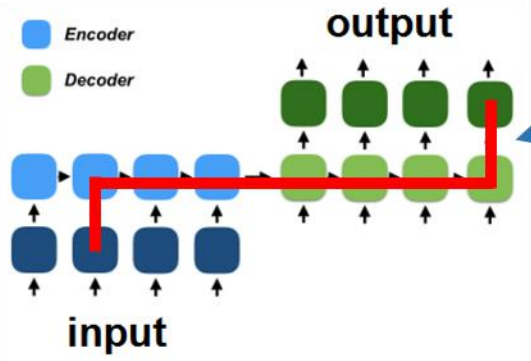



Tay Y, Dehghani M, Bahri D, et al. Efficient transformers: A survey[J]. arXiv preprint arXiv:2009.06732, 2020.

# If we apply the Transformer in LSTF problem …

| | ARIMA | Prophet | LSTM | Conv | Transformer |
|---|---|---|---|---|---|
| Long Input | ✓ | ✓ | × | ✓ | × |
| Long Output | × | × | × | × | × |
| Complexity / layer | O(L) | Not clear | $O(L*d^2)$ | $O(k*L*d)$ | $O(L^2*d)$ |
| Max Path | | | O(L) | $O(\log_k L)$ | O(1) |

**L** is sequence length, **D** is the representation dimension, **k** is the kernel size of Convs.




Path (between output and input)

The shorter path, the less signals traverse in BPTT.

# CONTENT

- Background
- Motivation: why self-attention?
- Methods: the details of Informer
- Experiments
- Summary

# Challenges

## The quadratic computation of self-attention.

The atom operation of self-attention mechanism, namely canonical dot-product, causes the time complexity and memory usage per layer to be O(L2).

[Complexity/layer]

## The memory bottleneck in stacking layers.

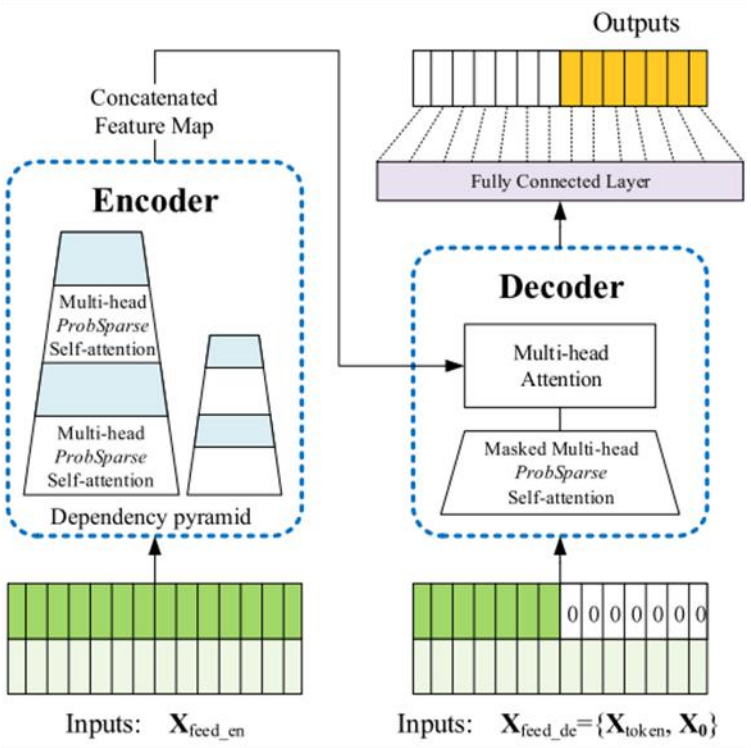
The stack of J encoder/decoder layer makes total memory usage to be O(J· L2), which limits the model scalability on receiving long sequence inputs.

[Long Input]

## The speed plunge in predicting long outputs.

The dynamic decoding of vanilla Transformer makes the inference speed as slow as RNN-based model.

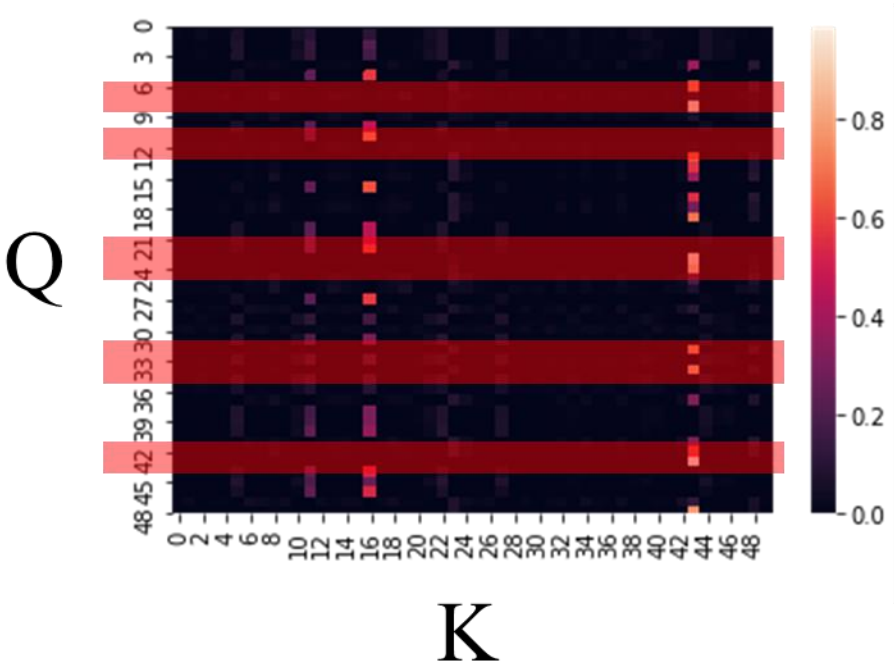
[Long Output]

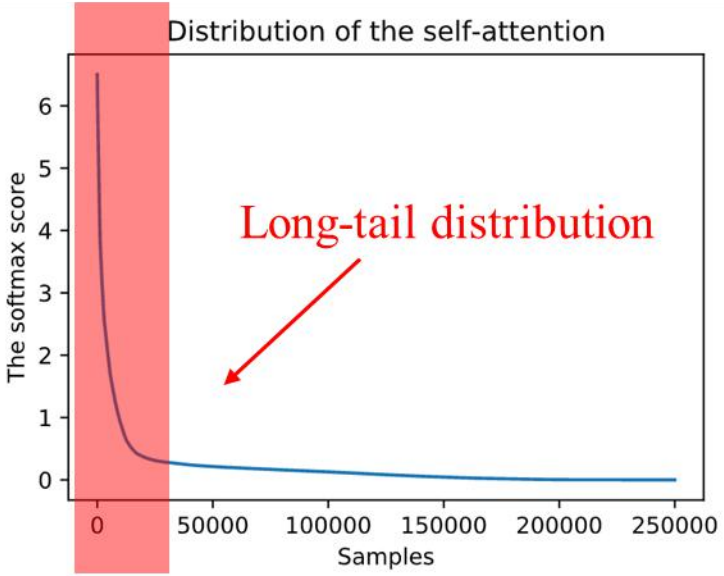



An overall graph of the Informer model.

# Challenge 1: Self-attention Mechanism

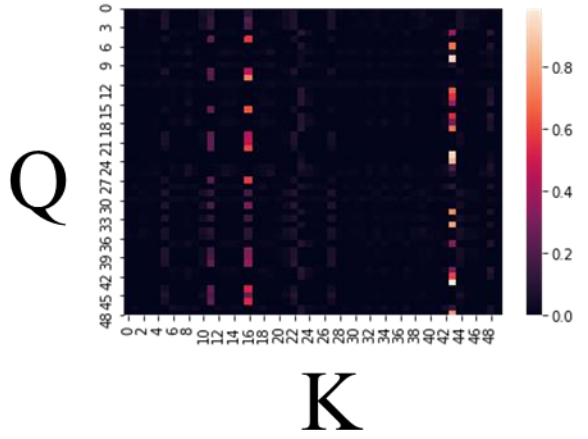
$$\mathcal{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d}})\mathbf{V}$$



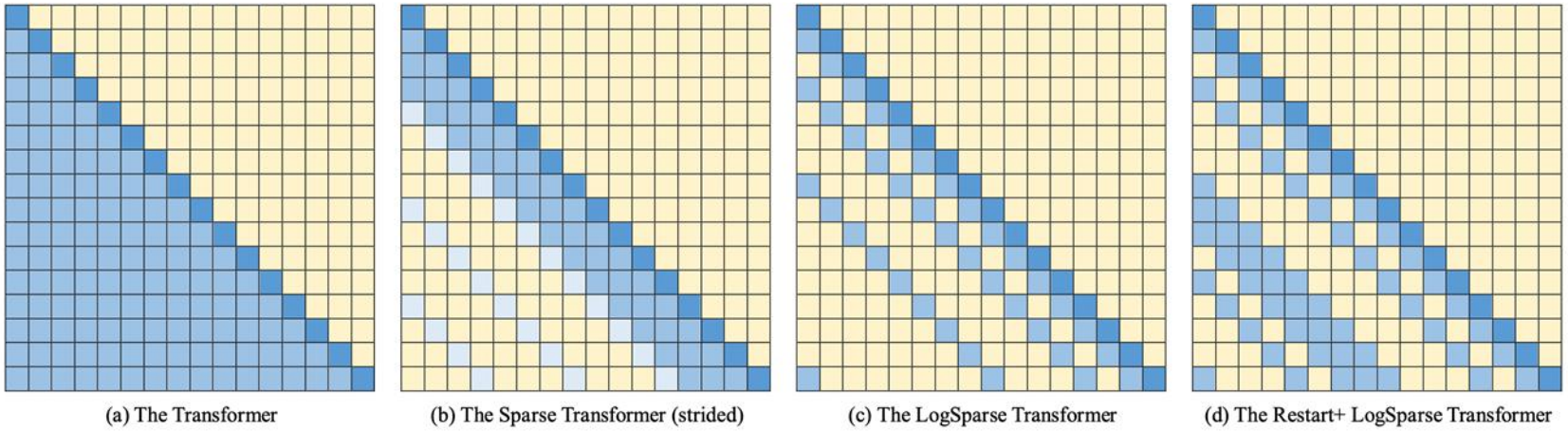
# Challenge 1: Self-attention Mechanism

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}$$




Previous work:

All heuristic!



(a) The Transformer (b) The Sparse Transformer (strided) (c) The LogSparse Transformer (d) The Restart+ LogSparse Transformer

# Challenge 1: Self-attention Mechanism

- Rewrite the original self-attention into the probability formulation

$$\mathcal{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V} \quad \Rightarrow \quad \mathcal{A}(\mathbf{q}_i,\mathbf{K},\mathbf{V}) = \sum_j \frac{k(\mathbf{q}_i,\mathbf{k}_j)}{\sum_l k(\mathbf{q}_i,\mathbf{k}_l)}\mathbf{v}_j = \mathbb{E}_{p(\mathbf{k}_j|\mathbf{q}_i)}[\mathbf{v}_j]$$

- Establish the measurement

Attention probability $p(\mathbf{k}_j|\mathbf{q}_i)$ ? Uniform probability $q(\mathbf{k}_j|\mathbf{q}_i) = \frac{1}{L_K}$

$$KL(q||p) = \sum_{j=1}^{L_K} \frac{1}{L_K} \ln \frac{1/L_K}{k(\mathbf{q}_i,\mathbf{k}_j)/\sum_l k(\mathbf{q}_i,\mathbf{k}_l)}$$

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

Dropping the constant, we define the *i*-th query's sparsity measurement as

$$M(\mathbf{q}_i,\mathbf{K}) = \ln \sum_{j=1}^{L_K} e^{\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}} - \frac{1}{L_K}\sum_{j=1}^{L_K}\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}$$

# Challenge 1: Self-attention Mechanism

- Rewrite the original self-attention into the probability formulation

$$\mathcal{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V} \quad \Rightarrow \quad \mathcal{A}(\mathbf{q}_i,\mathbf{K},\mathbf{V}) = \sum_j \frac{k(\mathbf{q}_i,\mathbf{k}_j)}{\sum_l k(\mathbf{q}_i,\mathbf{k}_l)}\mathbf{v}_j = \mathbb{E}_{p(\mathbf{k}_j|\mathbf{q}_i)}[\mathbf{v}_j]$$

- Establish the measurement

$$M(\mathbf{q}_i,\mathbf{K}) = \ln \sum_{j=1}^{L_K} e^{\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}} - \frac{1}{L_K}\sum_{j=1}^{L_K}\frac{\mathbf{q}_i\mathbf{k}_j^\top}{\sqrt{d}}$$

- Define *ProbSparse* self-attention

  Based on proposed measurement, we have the *ProbSparse* Self-attention by allowing each key only to attend to the *u* dominate queries

$$\mathcal{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Softmax}(\frac{\overline{\mathbf{Q}}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}$$

  where $\overline{\mathbf{Q}}$ is a sparse matrix of the same size of **q** and it only contains the Top-*u* queries under the sparsity measurement $M(\mathbf{q},\mathbf{K})$.

# Challenge 1: Self-attention Mechanism

- Rewrite the original self-attention into the probability formulation
- Establish the measurement
- Define *ProbSparse* self-attention
- The measurement have to calculate each dot-product pairs, which requires $O(L^2)$ computation. And log-sum-exp fun has the potential numerical stability issue.

$$M(\mathbf{q}_i, \mathbf{K}) = \ln \sum_{j=1}^{L_K} e^{\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}}} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \quad \textbf{Replace} \quad \max_j \{ \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \}$$

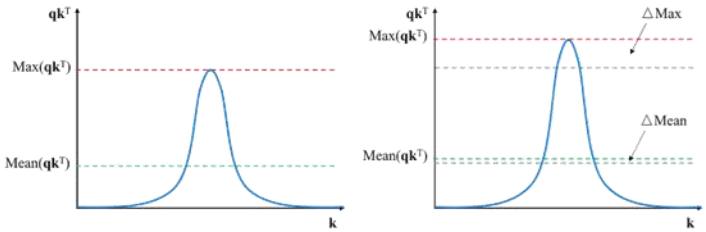
**Lemma** 1. *For each query* $\mathbf{q}_i \in \mathbb{R}^d$ *and* $\mathbf{k}_j \in \mathbb{R}^d$ *in the keys set* $\mathbf{K}$, *we have the following lower and upper bounds*

$$\ln L_K \leq M(\mathbf{q}_i, \mathbf{K}) \leq \max_j \{ \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \} - \frac{1}{L_K} \sum_{j=1}^{L_K} \{ \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \} + \ln L_K .$$

*Especially when* $\mathbf{q}_i \in \mathbf{K}$, *it also holds.*

# Challenge 1: Self-attention Mechanism

- Rewrite the original self-attention into the probability formulation
- Establish the measurement
- Define *ProbSparse* self-attention
- Use an approximation to the measurement




**Proposition** 1. *Assuming* $\mathbf{k_j} \sim \mathcal{N}(\mu, \Sigma)$ *and let* $\mathbf{qk}_i = \{\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} | j = 1, \ldots, L_K\}$, $\forall M_m = \max_i M(\mathbf{q}_i, \mathbf{K})$ *there exist* $\kappa > 0$ *such that in that interval* $\forall \mathbf{q}_1, \mathbf{q}_2 \in \{\mathbf{q} | M(\mathbf{q}, \mathbf{K}) \in [M_m, M_m - \kappa)\}$ *if* $\overline{M}(\mathbf{q}_1, \mathbf{K}) > \overline{M}(\mathbf{q}_2, \mathbf{K})$ *and* $\mathrm{Var}(\mathbf{qk}_1) > \mathrm{Var}(\mathbf{qk}_2)$, *we have high probability that* $M(\mathbf{q}_1, \mathbf{K}) > M(\mathbf{q}_2, \mathbf{K})$.

**Algorithm 1** ProbSparse self-attention

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

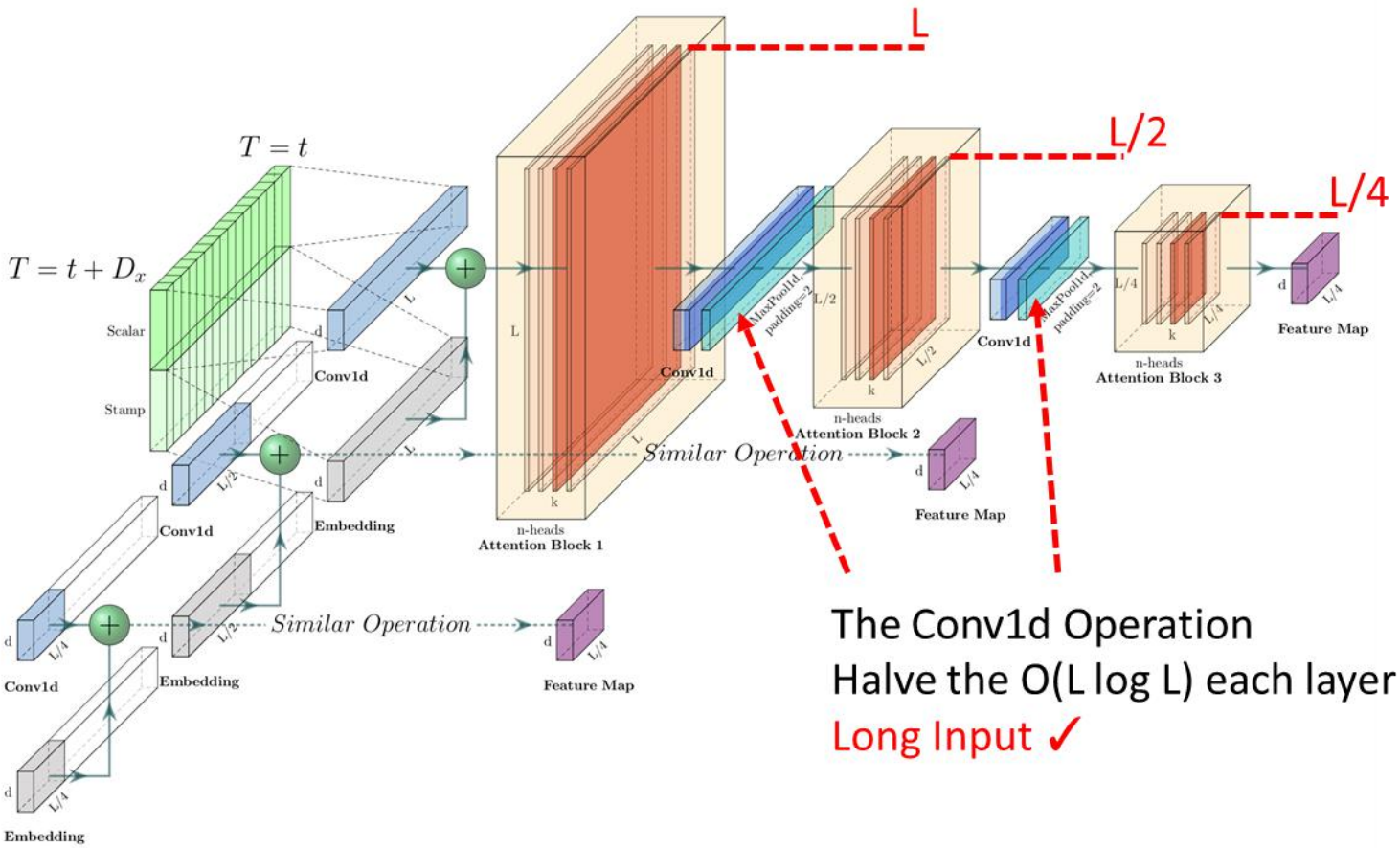
1: **initialize:** set hyperparameter $c$, $u = c \ln m$ and $U = m \ln n$
2: randomly select $U$ dot-product pairs from $\mathbf{K}$ as $\bar{\mathbf{K}}$
3: set the sample score $\bar{\mathbf{S}} = \mathbf{Q}\bar{\mathbf{K}}^\top$
4: compute the measurement $M = \max(\bar{\mathbf{S}}) - \mathrm{mean}(\bar{\mathbf{S}})$ by row
5: set Top-$u$ queries under $M$ as $\bar{\mathbf{Q}}$
6: set $\mathbf{S}_1 = \mathrm{softmax}(\bar{\mathbf{Q}}\mathbf{K}^\top/\sqrt{d}) \cdot \mathbf{V}$
7: set $\mathbf{S}_0 = \mathrm{mean}(\mathbf{V})$
8: set $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_0\}$ by their original rows accordingly

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

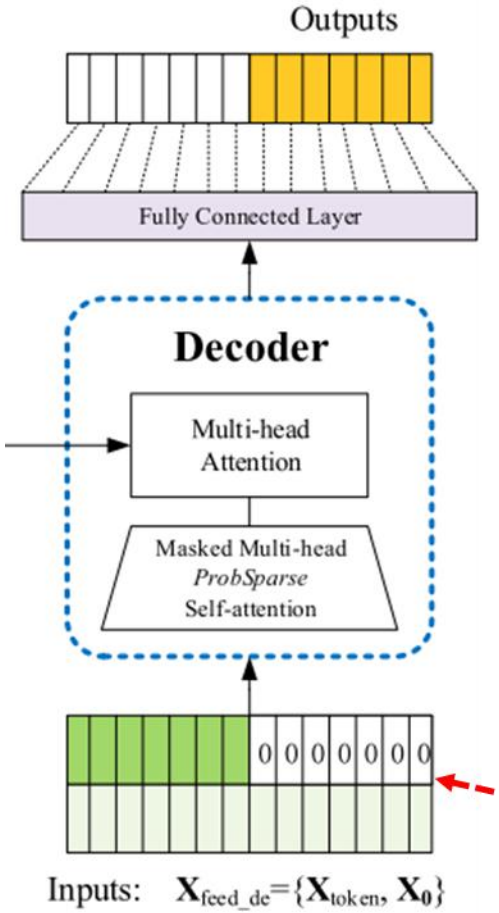
Sample L log L dot-product pairs → O(L log L)

$O(L^2)$ ✓

# Challenge 2: Self-attention Distilling Operation



# Challenge 3: Generative Style Decoder




Start token is an efficient technique in NLP "dynamic decoding", especially for per-training model, and we extend it into a generative way.

Instead of choosing a specific flag as the token, we sample a "shorter" long sequence in input sequence, which is an earlier slice before output sequence.

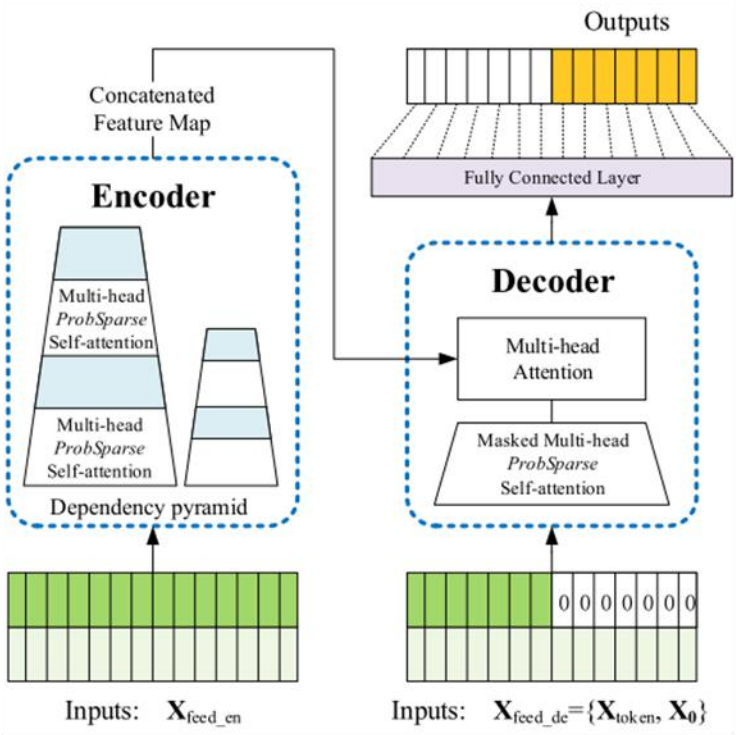
Take predicting 480 points as an example (5-day prediction in the ETT dataset), we will take the known 5 days before this time sequence as "start-token", and feed the generative-style inference decoder with them.

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

Long Output ✓

# The overall architecture of the proposed Informer model

**Challenges:** [Complexity/layer ✓][Long Input ✓][Long Output ✓]




An overall graph of the Informer model.

The Informer network components in details

| | | | |
|---|---|---|---|
| **Encoder:** | | | N |
| Inputs | 1x3 Conv1d | Embedding ($d = 512$) | 4 |
| ProbSparse Self-attention Block | Multi-head ProbSparse Attention ($h = 16$, $d = 32$) | | |
| | Add, LayerNorm, Dropout ($p = 0.1$) | | |
| | Pos-wise FFN ($d_{inner} = 2048$), GELU | | |
| | Add, LayerNorm, Dropout ($p = 0.1$) | | |
| Distilling | 1x3 conv1d, ELU | | |
| | Max pooling (stride = 2) | | |
| **Decoder:** | | | N |
| Inputs | 1x3 Conv1d | Embedding ($d = 512$) | 2 |
| Masked PSB | add Mask on Attention Block | | |
| Self-attention Block | Multi-head Attention ($h = 8$, $d = 64$) | | |
| | Add, LayerNorm, Dropout ($p = 0.1$) | | |
| | Pos-wise FFN ($d_{inner} = 2048$), GELU | | |
| | Add, LayerNorm, Dropout ($p = 0.1$) | | |
| **Final:** | | | |
| Outputs | FCN ($d = d_{out}$) | | |

# CONTENT

- Background
- Motivation: why self-attention?
- Methods: the details of Informer
- Experiments
- Summary

# Experiments settings

## Datasets

- ETT (Electricity Transformer Temperature)
  - 2 stations, 2 years, every 15 minutes. Train/Val/Test is 12/4/8 months.
- ECL (Electricity Consuming Load)
  - 1 country, 2 years, every 1 hour. Train/Val/Test is 15/3/6 months.
- Weather
  - 1600 US locations, 4 years, every 1 hour. Train/Val/Test is 28/10/10 months.

## Baselines

- Time-series methods: ARIMA, DeepAR, Prophet, LSTMa, LSTnet
- Transformer-based methods: Vanilla Transformer, Reformer, LogSparse Transformer

## Metrics

- Mean Absolute Error (MAE), Mean Squared Error (MSE)

## Platform

- All methods are run on one single Nvidia V100 GPU.

# Experiments results

## Overall Results: Univariate Time-series Forecasting

Table 1: Univariate long sequence time-series forecasting results on four datasets (five cases)

| Methods | Metric | $ETTh_1$ | | | | | $ETTh_2$ | | | | | $ETTm_1$ | | | | | Weather | | | | | ECL | | | | | count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 24 | 48 | 168 | 336 | 720 | 24 | 48 | 168 | 336 | 720 | 24 | 48 | 96 | 288 | 672 | 24 | 48 | 168 | 336 | 720 | 48 | 168 | 336 | 720 | 960 | |
| Informer | MSE | 0.062 | **0.108** | **0.146** | 0.208 | **0.193** | **0.079** | **0.103** | **0.143** | 0.171 | 0.184 | **0.051** | 0.092 | 0.119 | **0.181** | **0.204** | 0.107 | **0.164** | **0.226** | **0.241** | **0.259** | 0.335 | 0.408 | 0.451 | **0.466** | **0.470** | 28 |
| | MAE | 0.178 | 0.245 | **0.294** | 0.363 | 0.365 | **0.206** | **0.240** | **0.296** | 0.327 | 0.339 | **0.153** | 0.217 | 0.249 | **0.320** | **0.345** | 0.223 | **0.282** | **0.338** | **0.352** | **0.367** | 0.423 | 0.466 | 0.488 | **0.499** | **0.520** | |
| Informer† | MSE | **0.046** | 0.129 | 0.183 | **0.189** | 0.201 | 0.083 | 0.111 | 0.154 | **0.166** | **0.181** | 0.054 | **0.087** | **0.115** | 0.182 | 0.207 | 0.107 | 0.167 | 0.237 | 0.252 | 0.263 | 0.304 | 0.416 | 0.479 | 0.482 | 0.538 | 14 |
| | MAE | **0.152** | 0.274 | 0.337 | **0.346** | **0.357** | 0.213 | 0.249 | 0.306 | **0.323** | **0.338** | 0.160 | **0.210** | **0.248** | 0.323 | 0.353 | **0.220** | 0.284 | 0.352 | 0.366 | 0.374 | 0.404 | 0.478 | 0.508 | 0.515 | 0.560 | |
| LogTrans | MSE | 0.059 | 0.111 | 0.155 | 0.196 | 0.217 | 0.080 | 0.107 | 0.176 | 0.175 | 0.185 | 0.061 | 0.156 | 0.229 | 0.362 | 0.450 | 0.120 | 0.182 | 0.267 | 0.299 | 0.274 | 0.360 | 0.410 | 0.482 | 0.522 | 0.546 | 0 |
| | MAE | 0.191 | 0.263 | 0.309 | 0.370 | 0.379 | 0.221 | 0.262 | 0.344 | 0.345 | 0.349 | 0.192 | 0.322 | 0.397 | 0.512 | 0.582 | 0.247 | 0.312 | 0.387 | 0.416 | 0.387 | 0.455 | 0.481 | 0.521 | 0.551 | 0.563 | |
| Reformer | MSE | 0.172 | 0.228 | 1.460 | 1.728 | 1.948 | 0.235 | 0.434 | 0.961 | 1.532 | 1.862 | 0.055 | 0.229 | 0.854 | 0.962 | 1.605 | 0.197 | 0.268 | 0.590 | 1.692 | 1.887 | 0.917 | 1.635 | 3.448 | 4.745 | 6.841 | 0 |
| | MAE | 0.319 | 0.395 | 1.089 | 0.978 | 1.226 | 0.369 | 0.505 | 0.797 | 1.060 | 1.543 | 0.170 | 0.340 | 0.675 | 1.107 | 1.312 | 0.329 | 0.381 | 0.552 | 0.945 | 1.352 | 0.840 | 1.515 | 2.088 | 3.913 | 4.913 | |
| LSTMa | MSE | 0.094 | 0.175 | 0.210 | 0.556 | 0.635 | 0.135 | 0.172 | 0.359 | 0.516 | 0.562 | 0.099 | 0.289 | 0.255 | 0.480 | 0.988 | **0.107** | 0.166 | 0.305 | 0.404 | 0.784 | 0.475 | 0.703 | 1.186 | 1.473 | 1.493 | 1 |
| | MAE | 0.232 | 0.322 | 0.352 | 0.644 | 0.704 | 0.275 | 0.318 | 0.470 | 0.548 | 0.613 | 0.201 | 0.371 | 0.370 | 0.528 | 0.805 | 0.222 | 0.298 | 0.404 | 0.476 | 0.709 | 0.509 | 0.617 | 0.854 | 0.910 | 0.9260 | |
| DeepAR | MSE | 0.089 | 0.126 | 0.213 | 0.403 | 0.614 | 0.080 | 0.125 | 0.179 | 0.568 | 0.367 | 0.075 | 0.197 | 0.336 | 0.908 | 2.371 | 0.108 | 0.177 | 0.259 | 0.535 | 0.407 | **0.188** | **0.295** | **0.388** | 0.471 | 0.583 | 6 |
| | MAE | 0.242 | 0.291 | 0.382 | 0.496 | 0.643 | 0.229 | 0.283 | 0.346 | 0.555 | 0.488 | 0.205 | 0.332 | 0.450 | 0.739 | 1.256 | 0.242 | 0.313 | 0.397 | 0.580 | 0.506 | **0.317** | **0.398** | **0.471** | 0.507 | 0.583 | |
| ARIMA | MSE | 0.086 | 0.133 | 0.364 | 0.428 | 0.613 | 3.538 | 3.168 | 2.768 | 2.717 | 2.822 | 0.074 | 0.157 | 0.242 | 0.424 | 0.565 | 0.199 | 0.247 | 0.471 | 0.678 | 0.996 | 0.861 | 1.014 | 1.102 | 1.213 | 1.322 | 1 |
| | MAE | 0.190 | **0.242** | 0.456 | 0.537 | 0.684 | 0.407 | 0.440 | 0.555 | 0.680 | 0.952 | 0.168 | 0.274 | 0.357 | 0.500 | 0.605 | 0.321 | 0.375 | 0.541 | 0.666 | 0.853 | 0.726 | 0.797 | 0.834 | 0.883 | 0.908 | |
| Prophet | MSE | 0.093 | 0.150 | 1.194 | 1.509 | 2.685 | 0.179 | 0.284 | 2.113 | 2.052 | 3.287 | 0.102 | 0.117 | 0.146 | 0.414 | 2.671 | 0.280 | 0.421 | 2.409 | 1.931 | 3.759 | 0.506 | 2.711 | 2.220 | 4.201 | 6.827 | 0 |
| | MAE | 0.241 | 0.300 | 0.721 | 1.766 | 3.155 | 0.345 | 0.428 | 1.018 | 2.487 | 4.592 | 0.256 | 0.273 | 0.304 | 0.482 | 1.112 | 0.403 | 0.492 | 1.092 | 2.406 | 1.030 | 0.557 | 1.239 | 3.029 | 1.363 | 4.184 | |

- The *Informer* greatly improves the inference performance.
- The *Informer* and its canonical degradation *Informer+* show comparable performance and Informer beats *Informer+* mostly in wining-counts, i.e. 28>14.
- The *Informer* model shows significantly better results than time-series models.
- Our proposed method achieves better results than vTrans, Reformer and LogSparseTrans.

# Experiments results

## Overall Results: Multivariate Time-series Forecasting

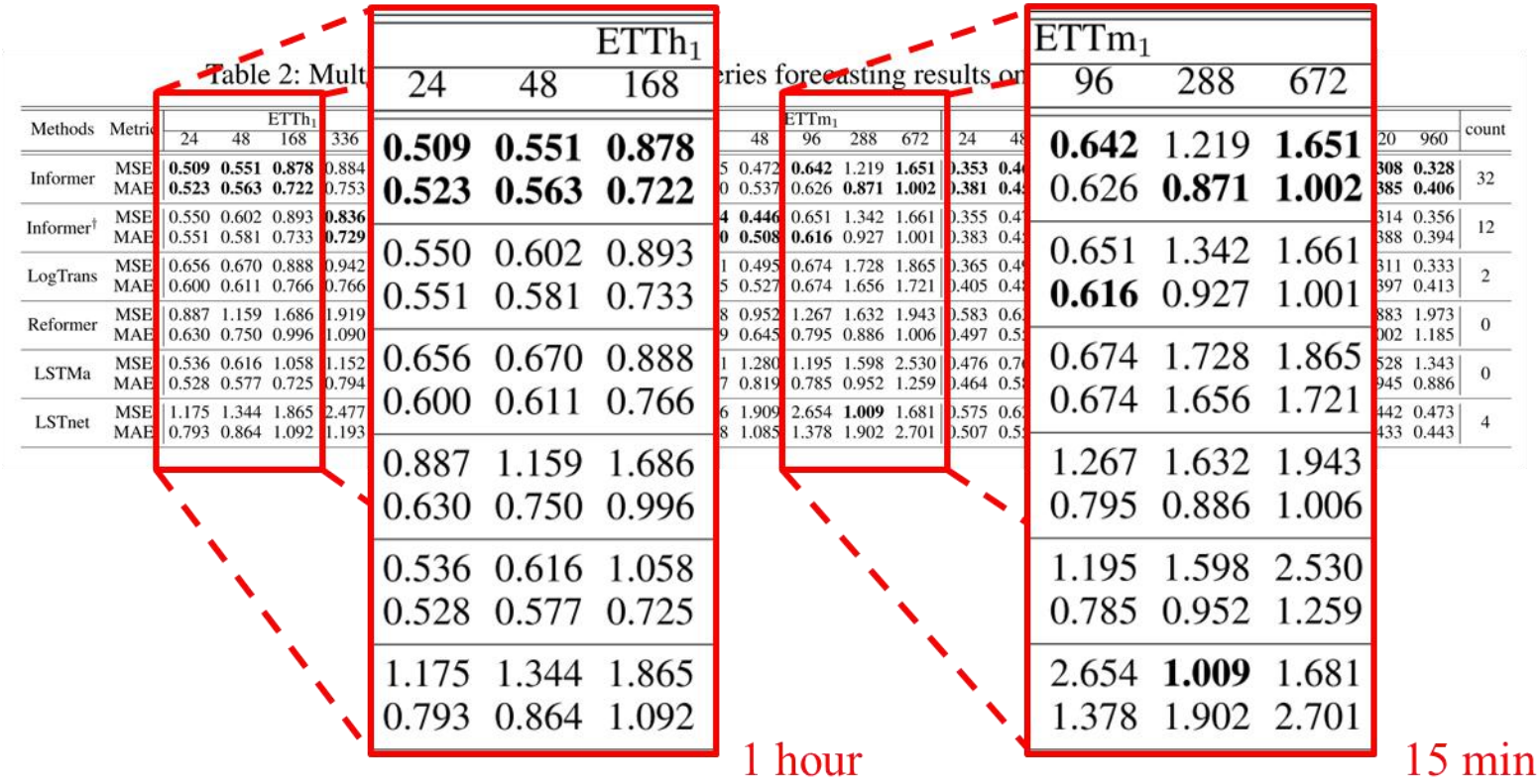
Table 2: Multivariate long sequence time-series forecasting results on four datasets (five cases)

| Methods | Metric | $ETTh_1$ | | | | | $ETTh_2$ | | | | | $ETTm_1$ | | | | | Weather | | | | | ECL | | | | | count |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 24 | 48 | 168 | 336 | 720 | 24 | 48 | 168 | 336 | 720 | 24 | 48 | 96 | 288 | 672 | 24 | 48 | 168 | 336 | 720 | 48 | 168 | 336 | 720 | 960 | |
| Informer | MSE | **0.509** | **0.551** | **0.878** | 0.884 | **0.941** | **0.446** | **0.934** | 1.512 | 1.665 | **2.340** | 0.325 | 0.472 | **0.642** | 1.219 | **1.651** | **0.353** | **0.464** | **0.592** | **0.623** | 0.685 | 0.269 | 0.300 | 0.311 | **0.308** | **0.328** | 32 |
| | MAE | **0.523** | **0.563** | **0.722** | 0.753 | **0.768** | **0.523** | **0.733** | **0.996** | **1.035** | **1.209** | 0.440 | 0.537 | 0.626 | **0.871** | **1.002** | **0.381** | **0.455** | **0.531** | **0.546** | 0.575 | 0.351 | 0.376 | 0.385 | **0.385** | **0.406** | |
| Informer† | MSE | 0.550 | 0.602 | 0.893 | **0.836** | 0.981 | 0.683 | 0.977 | 1.873 | 1.374 | 2.493 | **0.324** | **0.446** | 0.651 | 1.342 | 1.661 | 0.355 | 0.471 | 0.613 | 0.626 | **0.680** | 0.269 | **0.281** | 0.309 | 0.314 | 0.356 | 12 |
| | MAE | 0.551 | 0.581 | 0.733 | **0.729** | 0.779 | 0.637 | 0.793 | 1.094 | 0.935 | 1.253 | **0.440** | **0.508** | **0.616** | 0.927 | 1.001 | 0.383 | 0.456 | 0.544 | 0.548 | **0.569** | 0.351 | **0.366** | **0.383** | 0.388 | 0.394 | |
| LogTrans | MSE | 0.656 | 0.670 | 0.888 | 0.942 | 1.109 | 0.726 | 1.728 | 3.944 | 3.711 | 2.817 | 0.341 | 0.495 | 0.674 | 1.728 | 1.865 | 0.365 | 0.496 | 0.649 | 0.666 | 0.741 | **0.267** | 0.290 | **0.305** | 0.311 | 0.333 | 2 |
| | MAE | 0.600 | 0.611 | 0.766 | 0.766 | 0.843 | 0.638 | 0.944 | 1.573 | 1.587 | 1.356 | 0.495 | 0.527 | 0.674 | 1.656 | 1.721 | 0.405 | 0.485 | 0.573 | 0.584 | 0.611 | 0.366 | 0.382 | 0.395 | 0.397 | 0.413 | |
| Reformer | MSE | 0.887 | 1.159 | 1.686 | 1.919 | 2.177 | 1.381 | 1.715 | 4.484 | 3.798 | 5.111 | 0.598 | 0.952 | 1.267 | 1.632 | 1.943 | 0.583 | 0.633 | 1.228 | 1.770 | 2.548 | 1.312 | 1.453 | 1.507 | 1.883 | 1.973 | 0 |
| | MAE | 0.630 | 0.750 | 0.996 | 1.090 | 1.218 | 1.475 | 1.585 | 1.650 | 1.508 | 1.793 | 0.489 | 0.645 | 0.795 | 0.886 | 1.006 | 0.497 | 0.556 | 0.763 | 0.997 | 1.407 | 0.911 | 0.975 | 0.978 | 1.002 | 1.185 | |
| LSTMa | MSE | 0.536 | 0.616 | 1.058 | 1.152 | 1.682 | 1.049 | 1.331 | 3.987 | 3.276 | 3.711 | 0.511 | 1.280 | 1.195 | 1.598 | 2.530 | 0.476 | 0.763 | 0.948 | 1.497 | 1.314 | 0.388 | 0.492 | 0.778 | 1.528 | 1.343 | 0 |
| | MAE | 0.528 | 0.577 | 0.725 | 0.794 | 1.018 | 0.689 | 0.805 | 1.560 | 1.375 | 1.520 | 0.517 | 0.819 | 0.785 | 0.952 | 1.259 | 0.464 | 0.589 | 0.713 | 0.889 | 0.875 | 0.444 | 0.498 | 0.629 | 0.945 | 0.886 | |
| LSTnet | MSE | 1.175 | 1.344 | 1.865 | 2.477 | 1.925 | 2.632 | 3.487 | **1.442** | **1.372** | 2.403 | 1.856 | 1.909 | 2.654 | **1.009** | 1.681 | 0.575 | 0.622 | 0.676 | 0.714 | 0.773 | 0.279 | 0.318 | 0.357 | 0.442 | 0.473 | 4 |
| | MAE | 0.793 | 0.864 | 1.092 | 1.193 | 1.084 | 1.337 | 1.577 | 2.389 | 2.429 | 3.403 | 1.058 | 1.085 | 1.378 | 1.902 | 2.701 | 0.507 | 0.553 | 0.585 | 0.607 | 0.643 | **0.337** | 0.368 | 0.391 | 0.433 | 0.443 | |

- The *Informer* greatly improves the inference performance.
- The *Informer* and its canonical degradation *Informer+* show comparable performance and Informer beats *Informer+* mostly in wining-counts, i.e. 32>12.
- The previous conclusion in multivariate case holds.

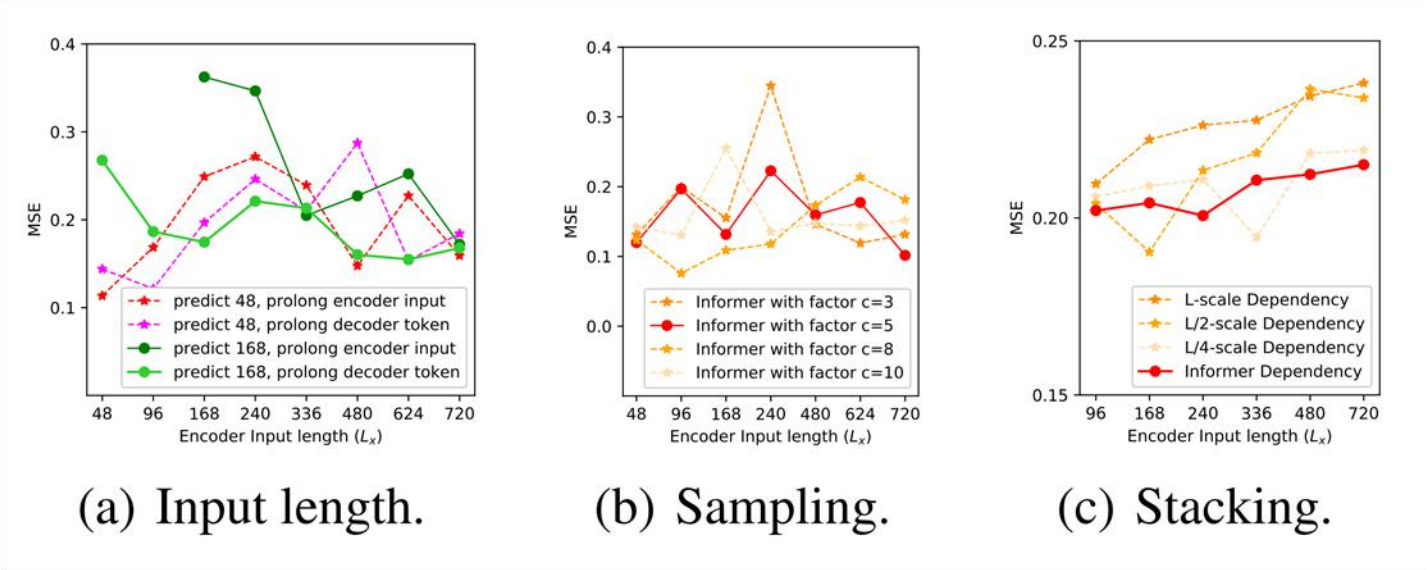
# Experiments results

## Overall Results: LSTF with Granularity Consideration



# Experiments results

## Parameter Sensitivity




(a) Input length. (b) Sampling. (c) Stacking.

# Experiments results

## Ablation study

Table 3: Ablation of *ProbSparse* mechanism

| Prediction length | | 336 | | | 720 | | |
|---|---|---|---|---|---|---|---|
| Encoder's input | | 336 | 720 | 1440 | 720 | 1440 | 2880 |
| Informer | MSE | 0.243 | 0.225 | 0.212 | 0.258 | 0.238 | 0.224 |
| | MAE | 0.487 | 0.404 | 0.381 | 0.503 | 0.399 | 0.387 |
| Informer† | MSE | 0.214 | 0.205 | - | 0.235 | - | - |
| | MAE | 0.369 | 0.364 | - | 0.401 | - | - |
| LogTrans | MSE | 0.256 | 0.233 | - | 0.264 | - | - |
| | MAE | 0.496 | 0.412 | - | 0.523 | - | - |
| Reformer | MSE | 1.848 | 1.832 | 1.817 | 2.094 | 2.055 | 2.032 |
| | MAE | 1.054 | 1.027 | 1.010 | 1.363 | 1.306 | 1.334 |

[1] Informer† uses the canonical self-attention mechanism.
[2] The '-' indicates failure for out-of-memory.

Table 4: Ablation of Self-attention Distilling

| Prediction length | | 336 | | | | | 480 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Encoder's input | | 336 | 480 | 720 | 960 | 1200 | 336 | 480 | 720 | 960 | 1200 |
| Informer† | MSE | 0.201 | 0.175 | 0.215 | 0.185 | 0.172 | 0.136 | 0.213 | 0.178 | 0.146 | 0.121 |
| | MAE | 0.360 | 0.335 | 0.366 | 0.355 | 0.321 | 0.282 | 0.382 | 0.345 | 0.296 | 0.272 |
| Informer‡ | MSE | 0.187 | 0.182 | 0.177 | - | - | 0.208 | 0.182 | 0.168 | - | - |
| | MAE | 0.330 | 0.341 | 0.329 | - | - | 0.384 | 0.337 | 0.304 | - | - |

[1] Informer‡ removes the self-attention distilling from Informer†.
[2] The '-' indicates failure for out-of-memory.

# Experiments results

## Ablation study

Table 5: Ablation of Generative Style Decoder

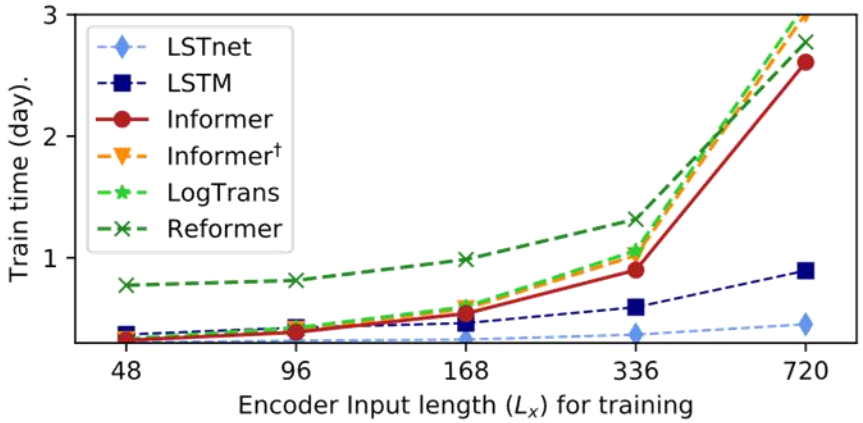
| Prediction length | | 336 | | | | 480 | | | |
|---|---|---|---|---|---|---|---|---|---|
| Prediction offset | | +0 | +12 | +24 | +48 | +0 | +48 | +96 | +168 |
| Informer‡ | MSE | 0.101 | 0.102 | 0.103 | 0.103 | 0.155 | 0.158 | 0.160 | 0.165 |
| | MAE | 0.215 | 0.218 | 0.223 | 0.227 | 0.317 | 0.397 | 0.399 | 0.406 |
| Informer§ | MSE | 0.152 | - | - | - | 0.462 | - | - | - |
| | MAE | 0.294 | - | - | - | 0.595 | - | - | - |

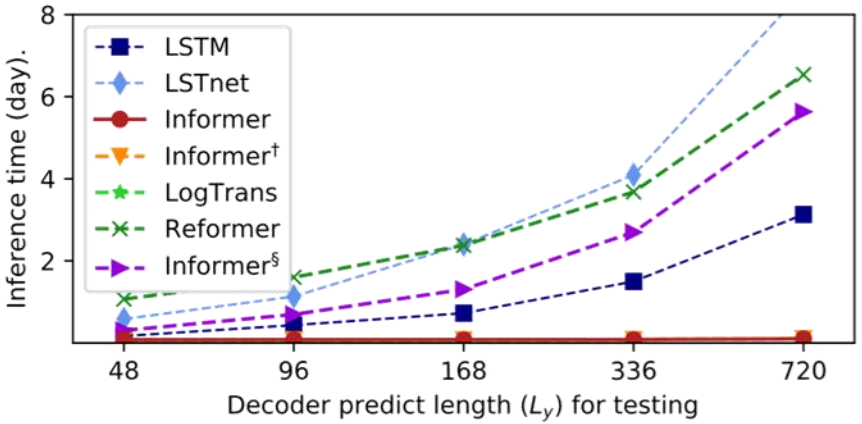
[1] Informer§ replaces our decoder with dynamic decoding one in Informer‡.

[2] The '-' indicates failure for the unacceptable metric results.

# Experiments results

## Computation efficiency




The total runtime of training/testing phase.

# CONTENT

- Background
- Motivation: why self-attention?
- Methods: the details of Informer
- Experiments
- Summary

# Things to take

- **Long Sequence Forecasting problem is a long-standing problem.**
- **The Self-attention model is a tractable solution in time-series problem.**

  *ProbSparse* Self-attention -> an efficient self-attention, also probabilistic/analyzable

  Self-attention Distilling Operation -> reduces overheads, allows for longer inputs

  Generative Style Decoder -> allows for longer outputs, even arbitrary-step predictions

- **Sparsity hypothesis works on Self-attention remarkably.**
- **Our proposed methods can bring substantial benefits to other domains, such as long sequence generation of text, music, image, and video.**

# We build a benchmark for the long sequence problem

- **Paper:** https://arxiv.org/abs/2012.07436
- **Code:** https://github.com/zhouhaoyi/Informer2020
- **Dataset:** https://github.com/zhouhaoyi/ETDataset

## Electricity Transformer Dataset (ETDataset)

In this Github repo, we provide several datasets could be used for the long sequence time-series problem. All datasets have been preprocessed and they were stored as `.csv` files. The dataset ranges from 2016/07 to 2018/07, and we will update to 2019 soon. 中文版本 | ChineseVersion

*Dataset list*

- **ETT-small**: The data of 2 Electricity Transformers at 2 stations, including load, oil temperature.
- **ETT-large**: The data of 39 Electricity Transformers at 39 stations, including load, oil temperature.
- **ETT-full**: The data of 69 Transformer station at 39 stations, including load, oil temperature, location, climate, demand.

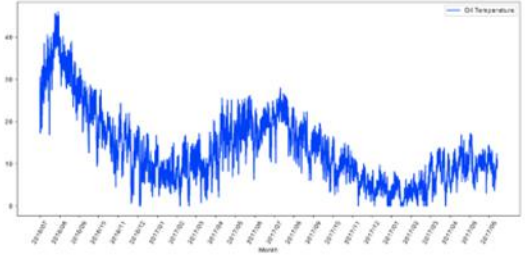


**Figure 1.**The overall view of "OT" in the ETT-small.

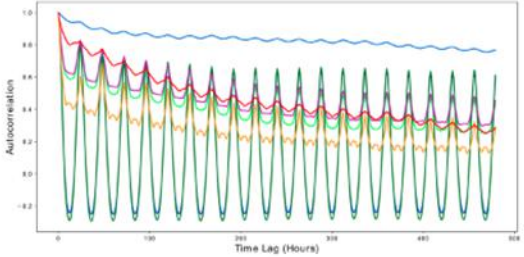



**Figure 2.**The autocorrelation graph of all variables.

北航ACTBD公众号

国际人工智能会议
AAAI 2021 论文北京预讲会

# THANKS

2020.12.19