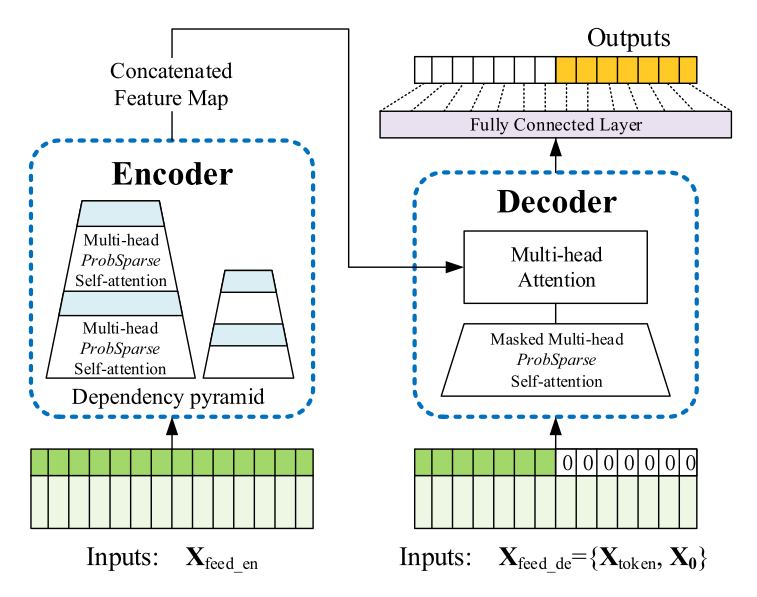
**https://github.com/zhouhaoyi/Informer2020**

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

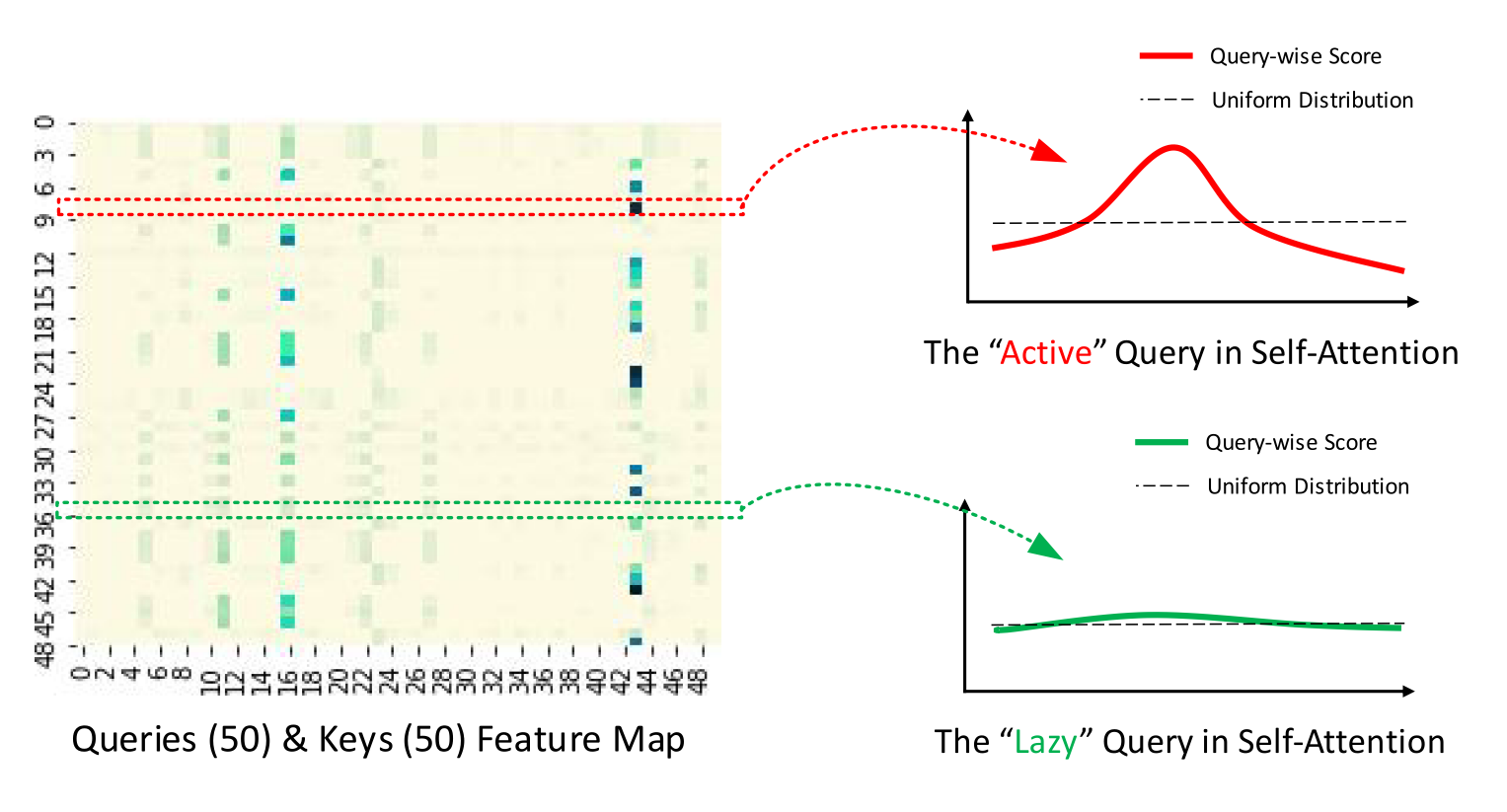
**Informer : فراتر از ترنسفورمر کارآمد برای پیش‌بینی سری‌های زمانی طولانی.**

This is the origin Pytorch implementation of Informer in the following paper: [Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting](https://arxiv.org/abs/2012.07436).

این پیاده‌سازی اصلی PyTorch از Informer بر اساس مقاله زیر است: **Informer: فراتر از ترنسفورمر کارآمد برای پیش‌بینی سری‌های زمانی طولانی.**

[](https://github.com/zhouhaoyi/Informer2020/blob/main/img/informer.png)  
  
**شکل 1- معماری اینفورمر**

توجه ProbSparse  
امتیازات خودتوجهی (Self-Attention) یک توزیع دم بلند تشکیل می‌دهند، جایی که کوئری‌های "فعال" در بخش "سر" امتیازات و کوئری‌های "تنبل" در بخش "دم" قرار دارند. ما توجه ProbSparse را طراحی کردیم تا کوئری‌های "فعال" را انتخاب کند نه کوئری‌های "تنبل". توجه ProbSparse با انتخاب کوئری‌های Top-u یک ترانسفورمر پراکنده را بر اساس توزیع احتمالات تشکیل می‌دهد. چرا به جای کوئری‌های Top-u از کلیدهای Top-u استفاده نکنیم؟ خروجی لایه خودتوجهی، بازنمایی مجدد ورودی است. این بازنمایی به‌صورت ترکیب وزنی مقادیر با توجه به امتیازات جفت‌های ضرب داخلی تعریف می‌شود. کوئری‌های برتر (Top queries) همراه با کلیدهای کامل، بازنمایی کاملی از مؤلفه‌های اصلی در ورودی را تشویق می‌کنند و این معادل انتخاب امتیازات "سر" در میان تمام جفت‌های ضرب داخلی است. اگر کلیدهای Top-u را انتخاب کنیم، کلیدهای کامل فقط جمع بی‌اهمیت مقادیر در میان امتیازات "دم بلند" را حفظ می‌کنند اما بازنمایی مؤلفه‌های اصلی را تخریب می‌کنند.

[](https://github.com/zhouhaoyi/Informer2020/blob/main/img/probsparse_intro.png)  
  
**شکل 2. تصویر ProbSparse Attention.**

**Requirements**

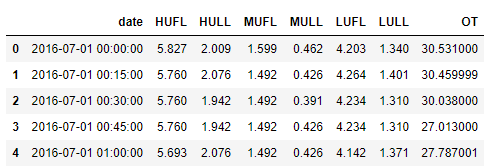
* Python 3.6
* matplotlib == 3.1.1
* numpy == 1.19.4
* pandas == 0.25.1
* scikit\_learn == 0.21.3
* torch == 1.8.0

Dependencies can be installed using the following command:

pip install -r requirements.txt

**داده ها**

**مجموعه داده ETT مورد استفاده در مقاله را می توان در repo ETDataset بارگیری کرد. فایل های داده مورد نیاز باید در پوشه data/ETT/ قرار داده شوند. یک برش نمایشی از داده های ETT در شکل زیر نشان داده شده است. توجه داشته باشید که ورودی هر مجموعه داده در این پیاده سازی به صورت میانگین صفر نرمال شده است.**

[](https://github.com/zhouhaoyi/Informer2020/blob/main/img/data.png)  
  
**شکل 3. نمونه ای از داده های ETT.**

**تکرارپذیری**

**برای بازتولید آسان نتایج می توانید مراحل زیر را دنبال کنید:**

**تصویر داکر را با استفاده از: make init راه اندازی کنید.**

**مجموعه داده ها را با استفاده از: make data دانلود کنید.**

**هر اسکریپت را با استفاده از make run\_module module="bash ETTh1.sh" برای هر اسکریپت اجرا کنید.**

**از طرف دیگر، تمام اسکریپت ها را یکجا اجرا کنید:**

استفاده

مثال‌های Colab: ما Google colabs را برای کمک به بازتولید و سفارشی‌سازی مخزن‌مان، که شامل آزمایش‌ها (آموزش و آزمایش)، پیش‌بینی، تجسم و داده‌های سفارشی است، ارائه می‌کنیم.

دستورات آموزش و آزمایش مدل با ProbSparse self-attention به ترتیب در مجموعه داده ETTh1، ETTh2 و ETTm1:

# ETTh1

python -u main\_informer.py --model informer --data ETTh1 --attn prob --freq h

# ETTh2

python -u main\_informer.py --model informer --data ETTh2 --attn prob --freq h

# ETTm1

python -u main\_informer.py --model informer --data ETTm1 --attn prob --freq t

اطلاعات بیشتر پارامتر لطفاً به main\_informer.py مراجعه کنید.

برای آموزش و آزمایش مدل، توضیحات دستوری دقیق‌تر و کامل‌تری ارائه می‌کنیم:

python -u main\_informer.py --model <model> --data <data>

--root\_path <root\_path> --data\_path <data\_path> --features <features>

--target <target> --freq <freq> --checkpoints <checkpoints>

--seq\_len <seq\_len> --label\_len <label\_len> --pred\_len <pred\_len>

--enc\_in <enc\_in> --dec\_in <dec\_in> --c\_out <c\_out> --d\_model <d\_model>

--n\_heads <n\_heads> --e\_layers <e\_layers> --d\_layers <d\_layers>

--s\_layers <s\_layers> --d\_ff <d\_ff> --factor <factor> --padding <padding>

--distil --dropout <dropout> --attn <attn> --embed <embed> --activation <activation>

--output\_attention --do\_predict --mix --cols <cols> --itr <itr>

--num\_workers <num\_workers> --train\_epochs <train\_epochs>

--batch\_size <batch\_size> --patience <patience> --des <des>

--learning\_rate <learning\_rate> --loss <loss> --lradj <lradj>

--use\_amp --inverse --use\_gpu <use\_gpu> --gpu <gpu> --use\_multi\_gpu --devices <devices>

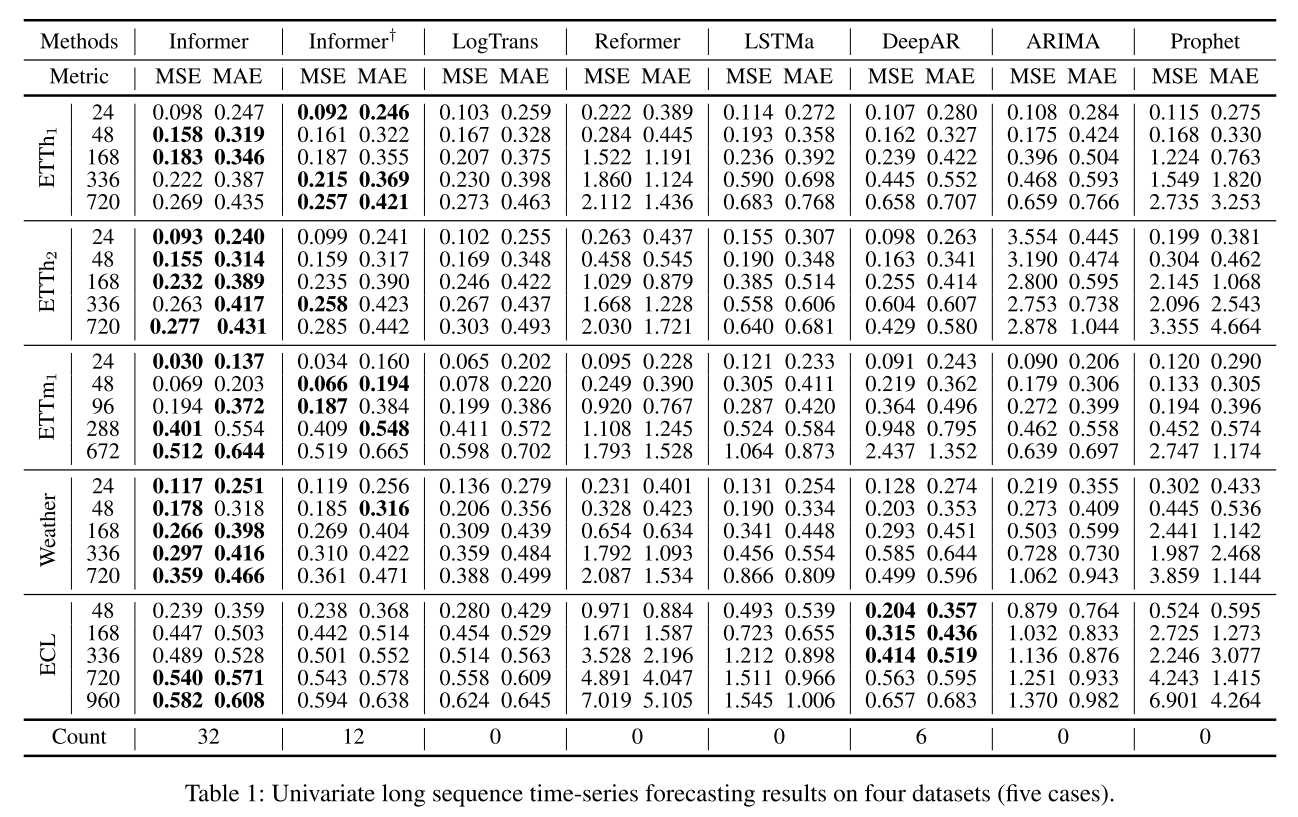
The detailed descriptions about the arguments are as following:

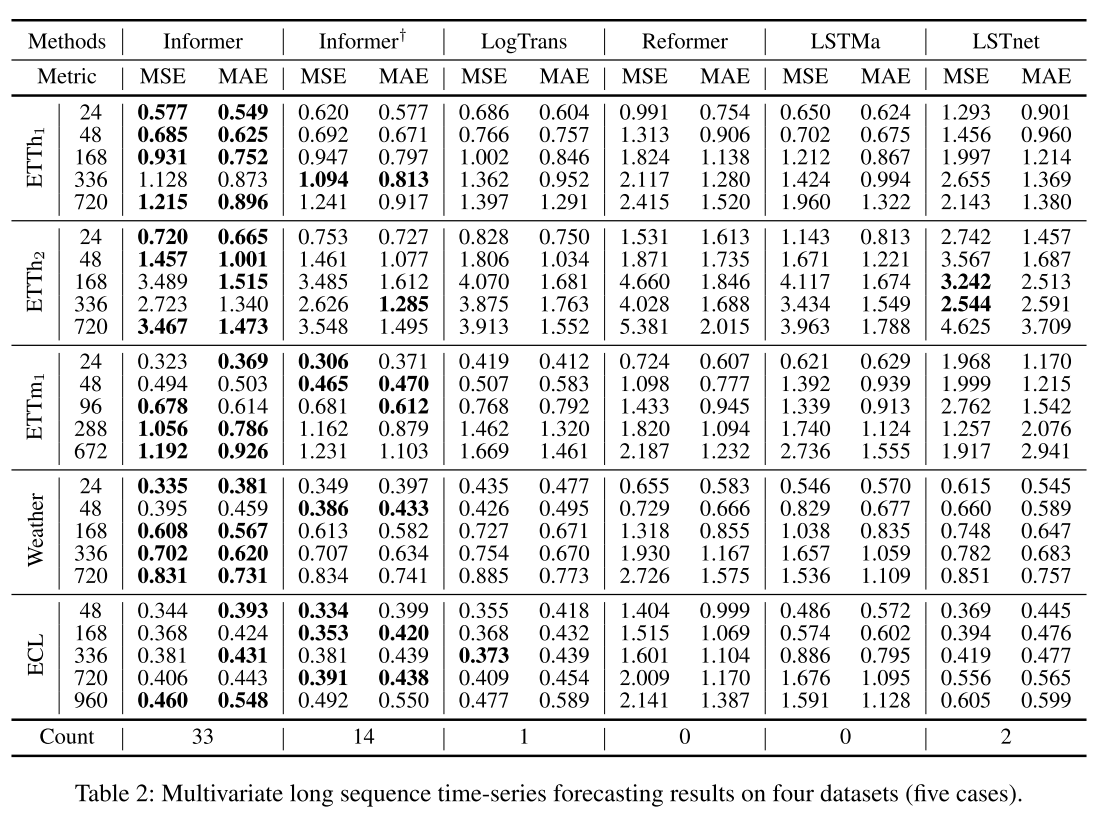
| **Parameter name** | **Description of parameter** |
| --- | --- |
| model | The model of experiment. This can be set to informer, informerstack, informerlight(TBD) |
| data | The dataset name |
| root\_path | The root path of the data file (defaults to ./data/ETT/) |
| data\_path | The data file name (defaults to ETTh1.csv) |
| features | The forecasting task (defaults to M). This can be set to M,S,MS (M : multivariate predict multivariate, S : univariate predict univariate, MS : multivariate predict univariate) |
| target | Target feature in S or MS task (defaults to OT) |
| freq | Freq for time features encoding (defaults to h). This can be set to s,t,h,d,b,w,m (s:secondly, t:minutely, h:hourly, d:daily, b:business days, w:weekly, m:monthly).You can also use more detailed freq like 15min or 3h |
| checkpoints | Location of model checkpoints (defaults to ./checkpoints/) |
| seq\_len | Input sequence length of Informer encoder (defaults to 96) |
| label\_len | Start token length of Informer decoder (defaults to 48) |
| pred\_len | Prediction sequence length (defaults to 24) |
| enc\_in | Encoder input size (defaults to 7) |
| dec\_in | Decoder input size (defaults to 7) |
| c\_out | Output size (defaults to 7) |
| d\_model | Dimension of model (defaults to 512) |
| n\_heads | Num of heads (defaults to 8) |
| e\_layers | Num of encoder layers (defaults to 2) |
| d\_layers | Num of decoder layers (defaults to 1) |
| s\_layers | Num of stack encoder layers (defaults to 3,2,1) |
| d\_ff | Dimension of fcn (defaults to 2048) |
| factor | Probsparse attn factor (defaults to 5) |
| padding | Padding type(defaults to 0). |
| distil | Whether to use distilling in encoder, using this argument means not using distilling (defaults to True) |
| dropout | The probability of dropout (defaults to 0.05) |
| attn | Attention used in encoder (defaults to prob). This can be set to prob (informer), full (transformer) |
| embed | Time features encoding (defaults to timeF). This can be set to timeF, fixed, learned |
| activation | Activation function (defaults to gelu) |
| output\_attention | Whether to output attention in encoder, using this argument means outputing attention (defaults to False) |
| do\_predict | Whether to predict unseen future data, using this argument means making predictions (defaults to False) |
| mix | Whether to use mix attention in generative decoder, using this argument means not using mix attention (defaults to True) |
| cols | Certain cols from the data files as the input features |
| num\_workers | The num\_works of Data loader (defaults to 0) |
| itr | Experiments times (defaults to 2) |
| train\_epochs | Train epochs (defaults to 6) |
| batch\_size | The batch size of training input data (defaults to 32) |
| patience | Early stopping patience (defaults to 3) |
| learning\_rate | Optimizer learning rate (defaults to 0.0001) |
| des | Experiment description (defaults to test) |
| loss | Loss function (defaults to mse) |
| lradj | Ways to adjust the learning rate (defaults to type1) |
| use\_amp | Whether to use automatic mixed precision training, using this argument means using amp (defaults to False) |
| inverse | Whether to inverse output data, using this argument means inversing output data (defaults to False) |
| use\_gpu | Whether to use gpu (defaults to True) |
| gpu | The gpu no, used for training and inference (defaults to 0) |
| use\_multi\_gpu | Whether to use multiple gpus, using this argument means using mulitple gpus (defaults to False) |
| devices | Device ids of multile gpus (defaults to 0,1,2,3) |

**Results**

We have updated the experiment results of all methods due to the change in data scaling. We are lucky that Informer gets performance improvement. Thank you @lk1983823 for reminding the data scaling in [issue 41](https://github.com/zhouhaoyi/Informer2020/issues/41).

Besides, the experiment parameters of each data set are formated in the .sh files in the directory ./scripts/. You can refer to these parameters for experiments, and you can also adjust the parameters to obtain better mse and mae results or draw better prediction figures.

[](https://github.com/zhouhaoyi/Informer2020/blob/main/img/result_univariate.png)  
  
**Figure 4.** Univariate forecasting results.

[](https://github.com/zhouhaoyi/Informer2020/blob/main/img/result_multivariate.png)  
  
**Figure 5.** Multivariate forecasting results.