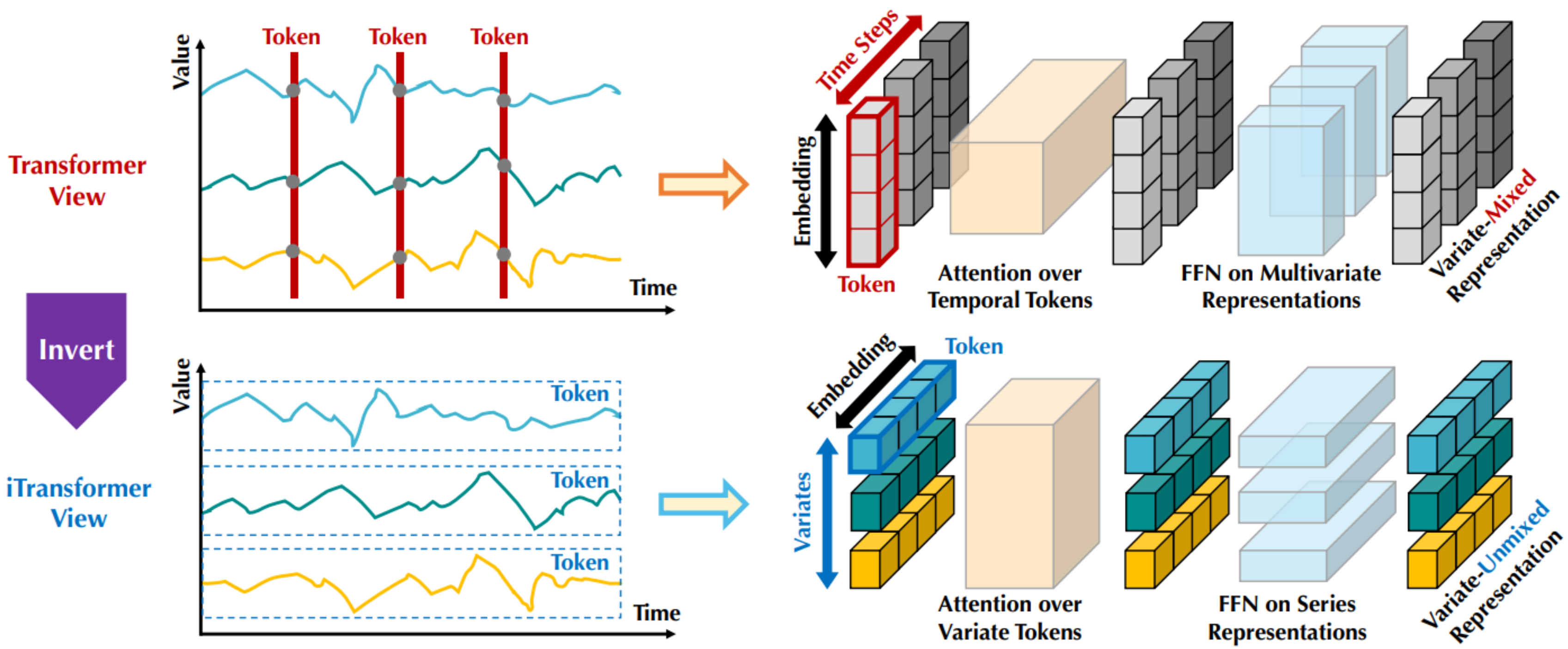
iTransformer

Inverted **Transformer** are effective for time series forcasting



预印本

iTransformer

Inverted **Transformer** are effective for time series forcasting

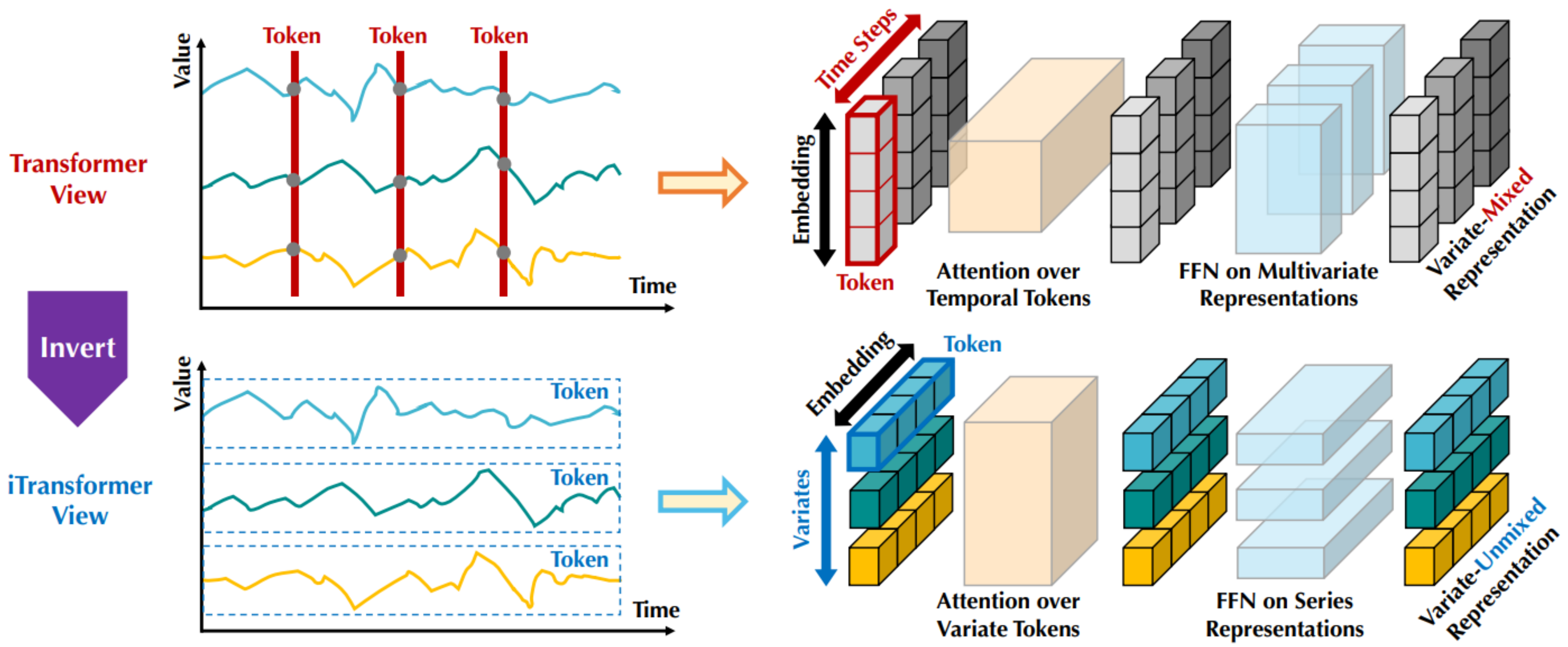
23.12.19

Presented by Yyyq

01 问题描述

线性模型 质疑了transformer在多变量时序预测任务上的有效性

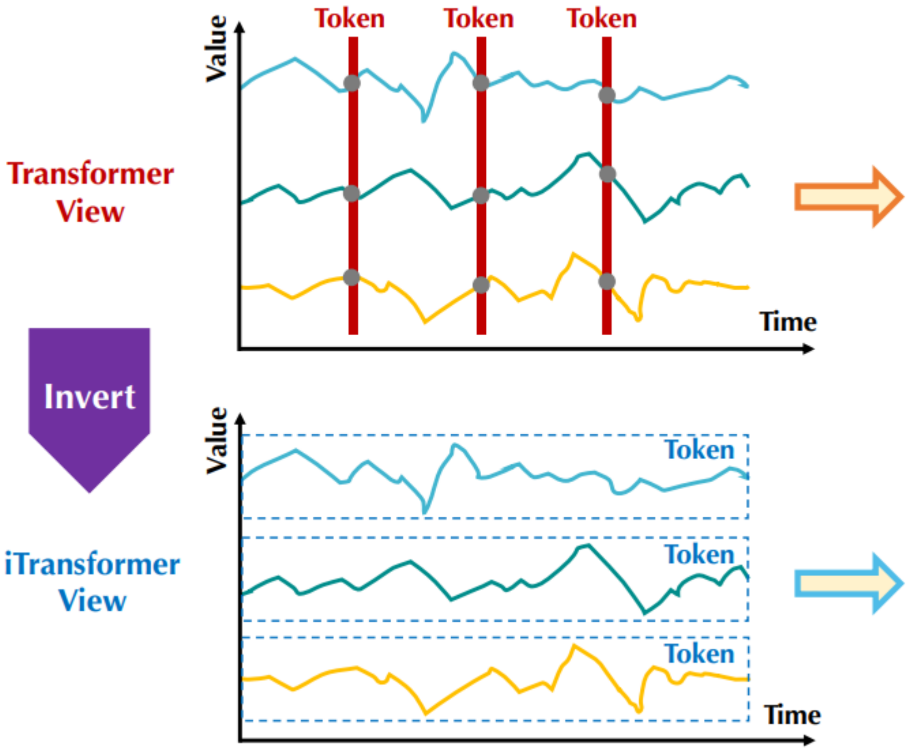
本篇文章将其归咎于——**Token的不恰当使用**

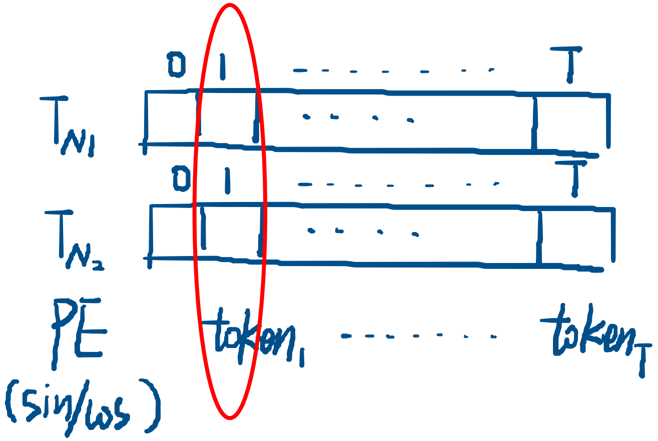


01 问题描述

线性模型 质疑了transformer在多变量时序预测任务上的有效性

本篇文章将其归咎于——**Token的不恰当使用**





Batchsize 变量 Node 序列长度 (Ex. 12)

(B, N, T)

"Token"

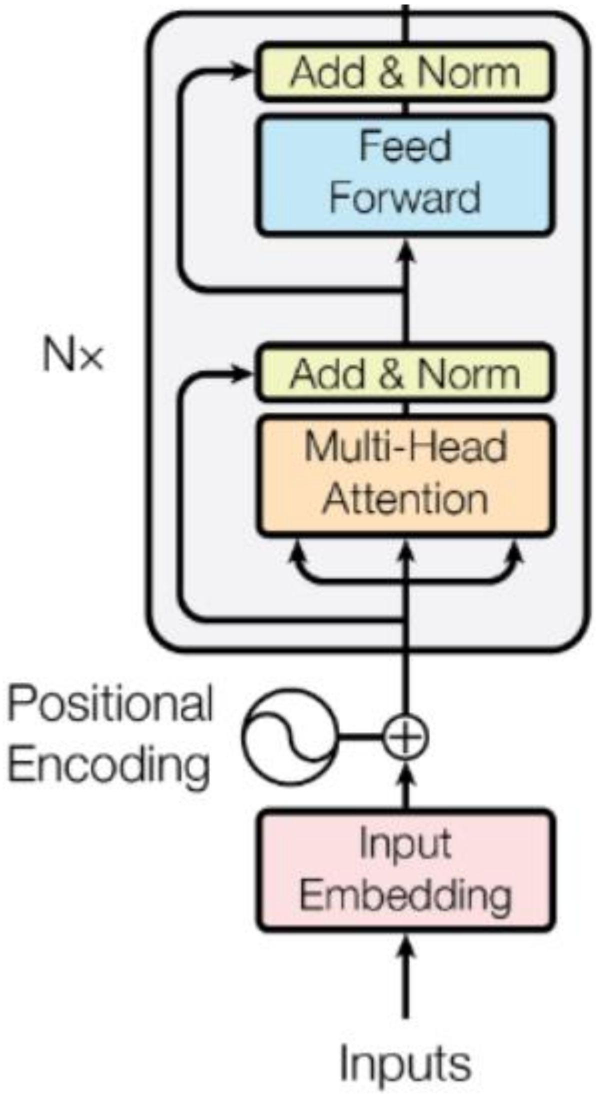
(B, N, T)

↓

(B, N, D)

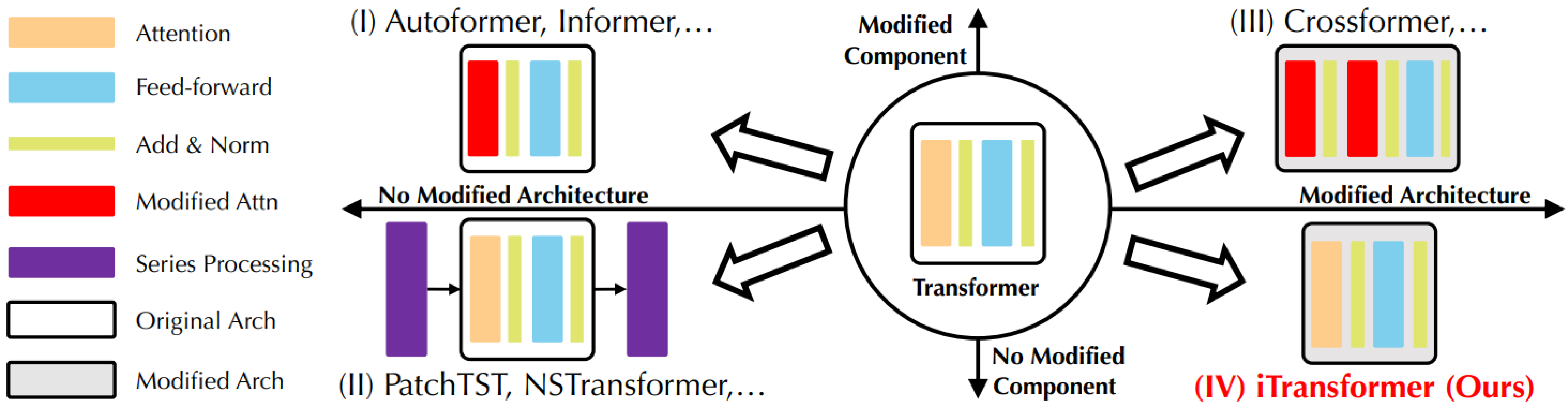
02 **创新点**

- **将独立的时间序列视为 Token**
- **注意力模块 和 前馈网络FFN 职责倒置**
  - 通过自注意力：捕获多变量相关性
  - 层归一化和FFN：学习 序列-全局 表示

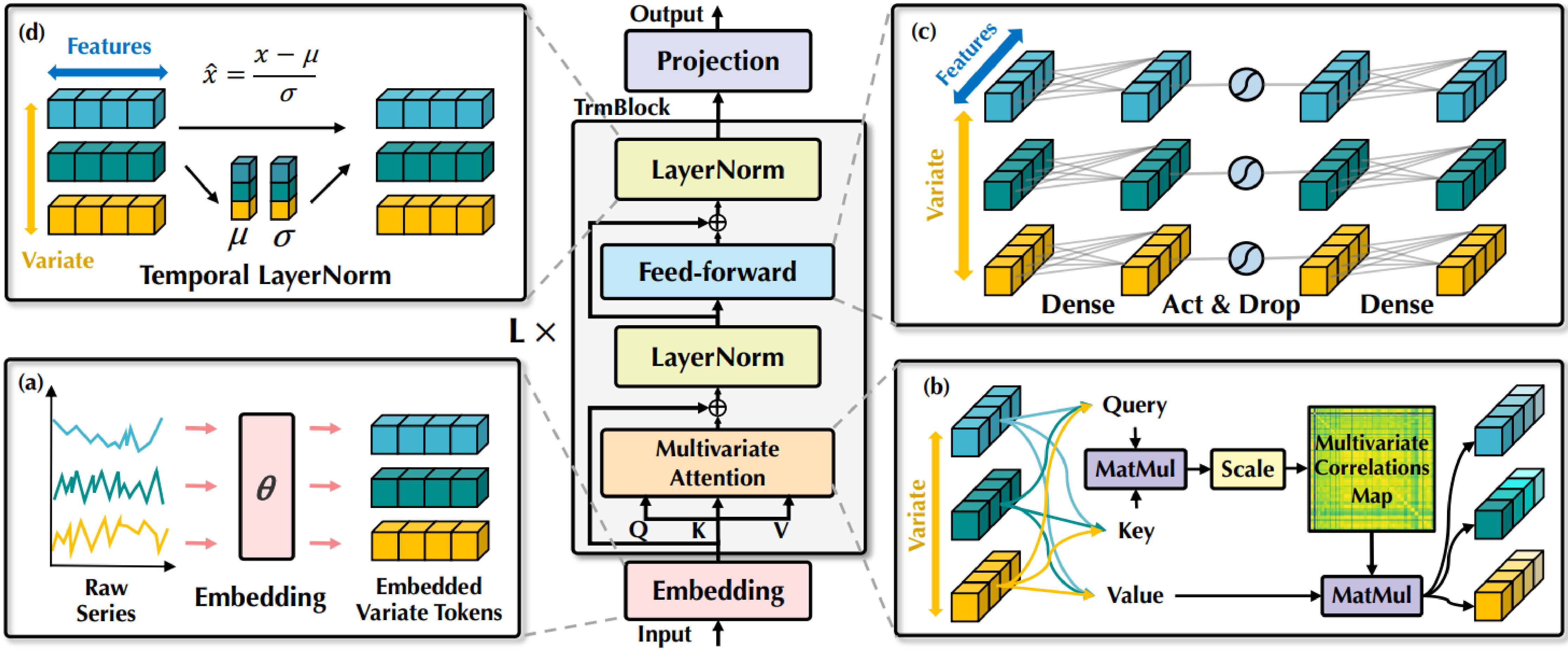


03 相关工作

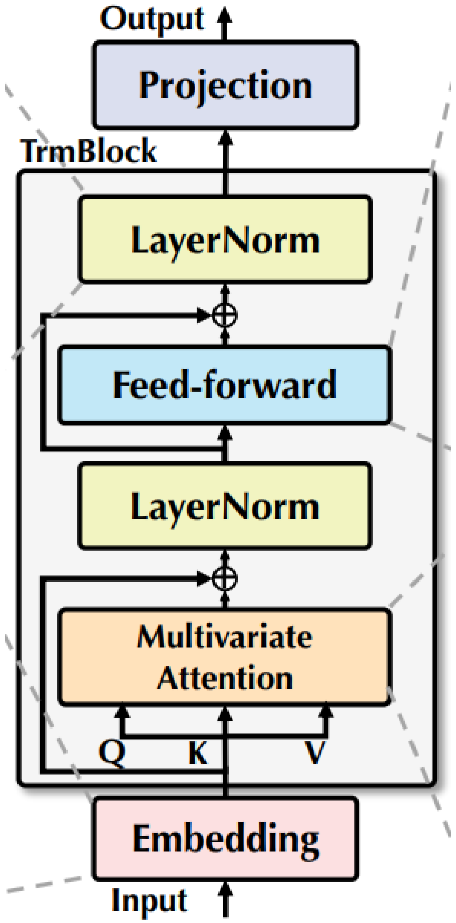
改进transformer模型：改框架？改模块？



04 iTransformer：Overview



04 iTransformer: *encoder-only*

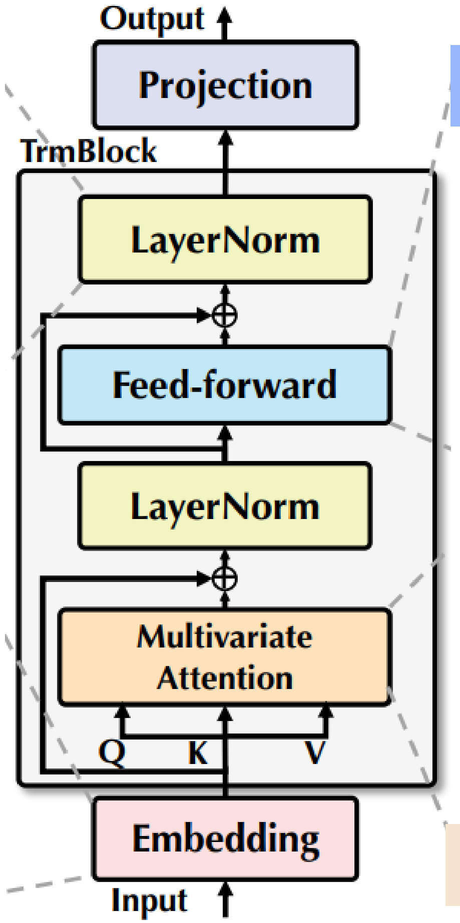


$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\} \in \mathbb{R}^{T \times N}$$

$$\begin{aligned}
\mathbf{h}_n^0 &= \mathrm{Embedding}(\mathbf{X}_{:,n}), \\
\mathbf{H}^{l+1} &= \mathrm{TrmBlock}(\mathbf{H}^l),\ l = 0, \cdots, L-1, \\
\hat{\mathbf{Y}}_{:,n} &= \mathrm{Projection}(\mathbf{h}_n^L),
\end{aligned}$$

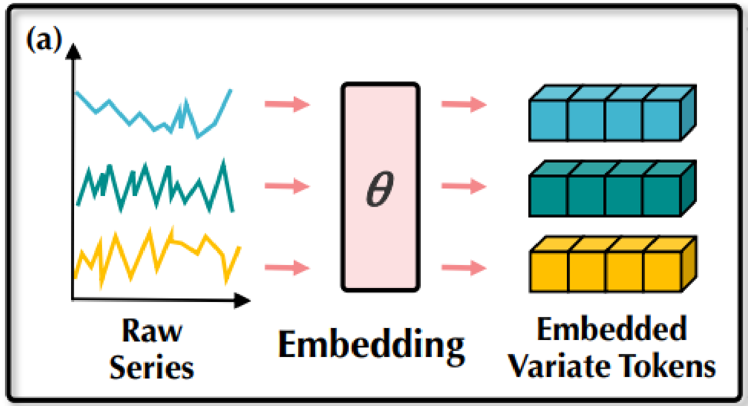
$$\mathbf{Y} = \{\mathbf{x}_{T+1}, \ldots, \mathbf{x}_{T+S}\} \in \mathbb{R}^{S \times N}.$$

iTransformer：Embedding



$\mathbb{R}^D \mapsto \mathbb{R}^S$

$\mathbb{R}^T \mapsto \mathbb{R}^D$




```
class DataEmbedding_inverted(nn.Module):
    def __init__(self, c_in, d_model, embed_type='fixed', freq='h', dropout=0.1):
        super(DataEmbedding_inverted, self).__init__()
        self.value_embedding = nn.Linear(c_in, d_model)
        self.dropout = nn.Dropout(p=dropout)

    def forward(self, x, x_mark):
        x = x.permute(0, 2, 1)
        # x: [Batch Variate Time]
        if x_mark is None:
            x = self.value_embedding(x)
        else:
            x = self.value_embedding(torch.cat([x, x_mark.permute(0, 2, 1)], 1))
        # x: [Batch Variate d_model]
        return self.dropout(x)
```

(B, T, N)

(B, N, T)

(B, N, D)

多变量

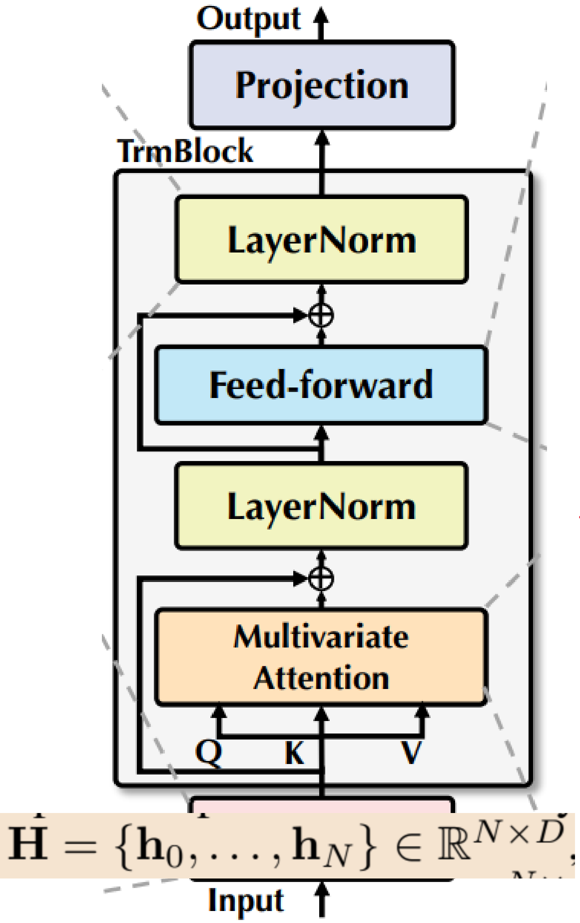
单特征

(B, N, T, 1)

(B, N, T, D)

token⊕

04 iTransformer：Self-attention



$$\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_k},$$

$$\mathbf{A}_{i,j} = (\mathbf{Q}\mathbf{K}^\top / \sqrt{d_k})_{i,j} \propto \mathbf{q}_i^\top \mathbf{k}_j.$$

$$\mathbf{A} \in \mathbb{R}^{N \times N}$$

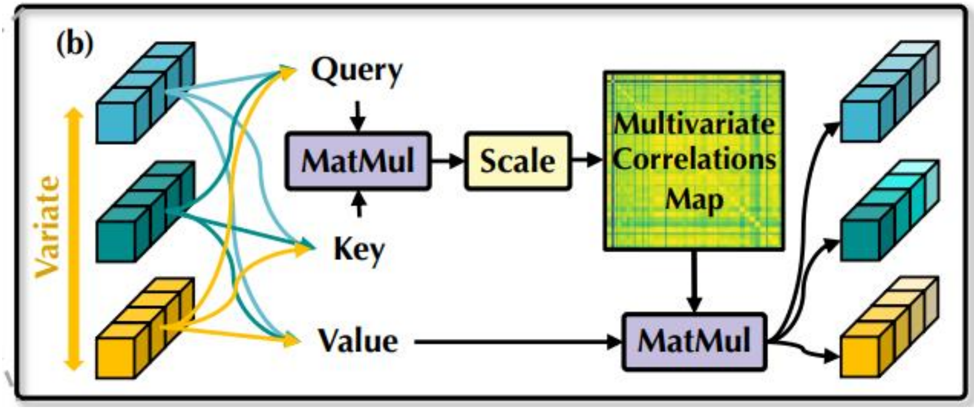
(B, N, D)

→ (B, N, h, dmodel)

A: N x N

多变量之间，自注意力分数

捕获多变量相关性



B, N, T, D

→ B, N, h, T, dmodel

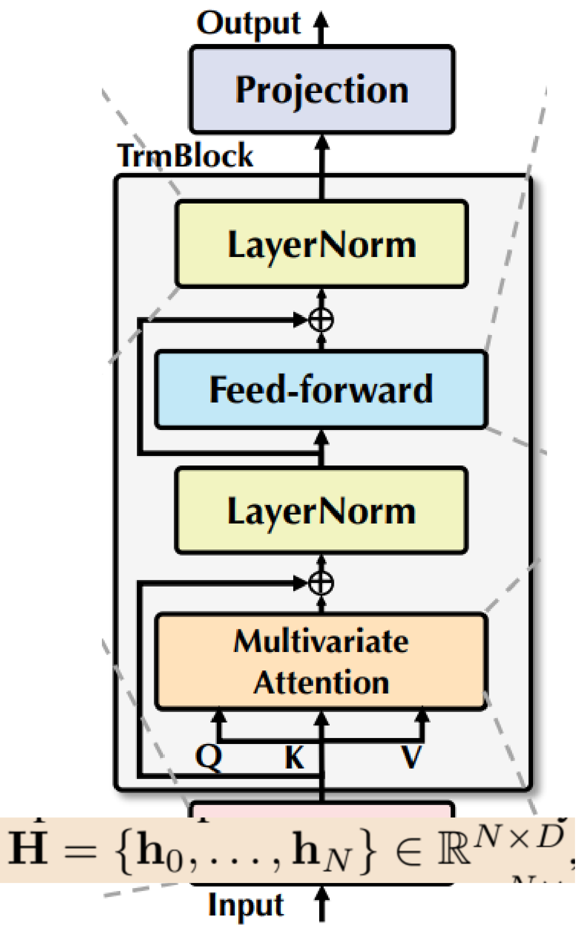
head

A: TxT

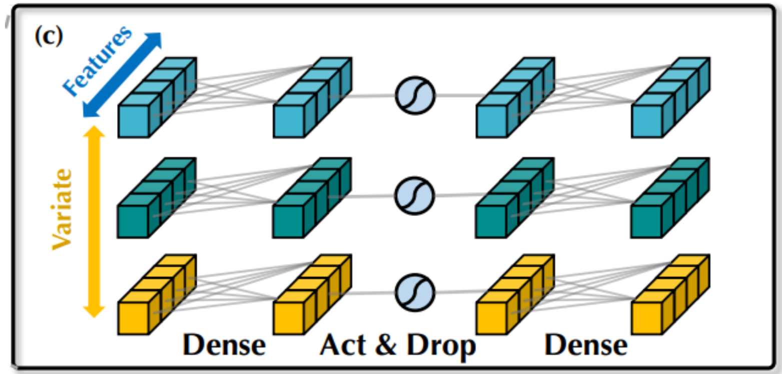
序列内，自注意力分数

捕获序列内部相关性

04 iTransformer：Feed-Forward Network



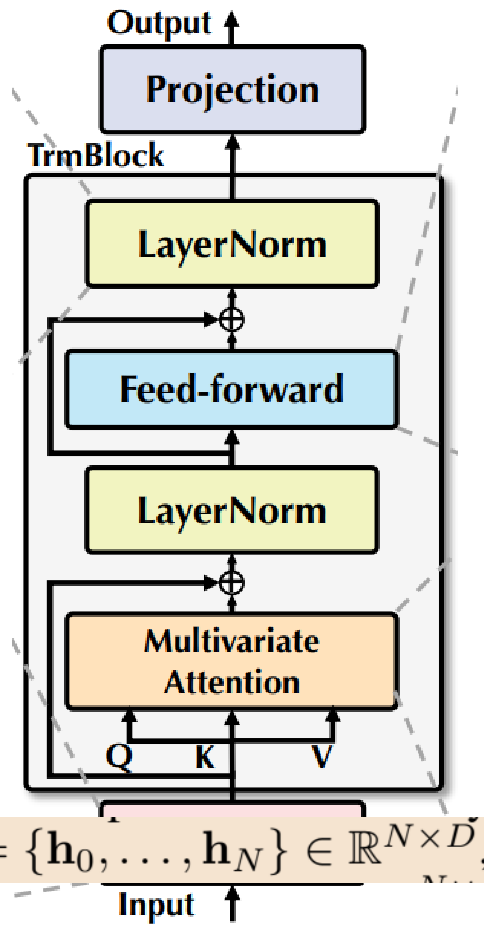
FFN层：激活函数、非线性映射（Conv1d×2）



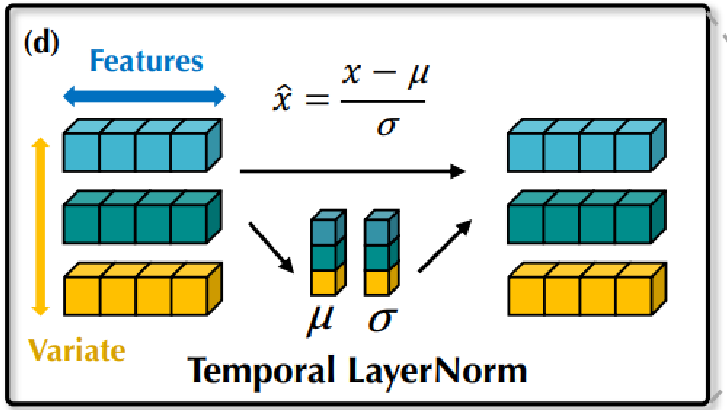
Conv1d (1)：对历史时间序列编码 $d_{model} \rightarrow d_{ff}$

Conv1d (2)：解码未来序列 $d_{ff} \rightarrow d_{model}$

04 iTransformer：Layer Normalization



$\mathbf{H}=\{\mathbf{h}_0,\ldots,\mathbf{h}_N\}\in\mathbb{R}^{N\times D}$,



归一化：应用于单变量的序列表示

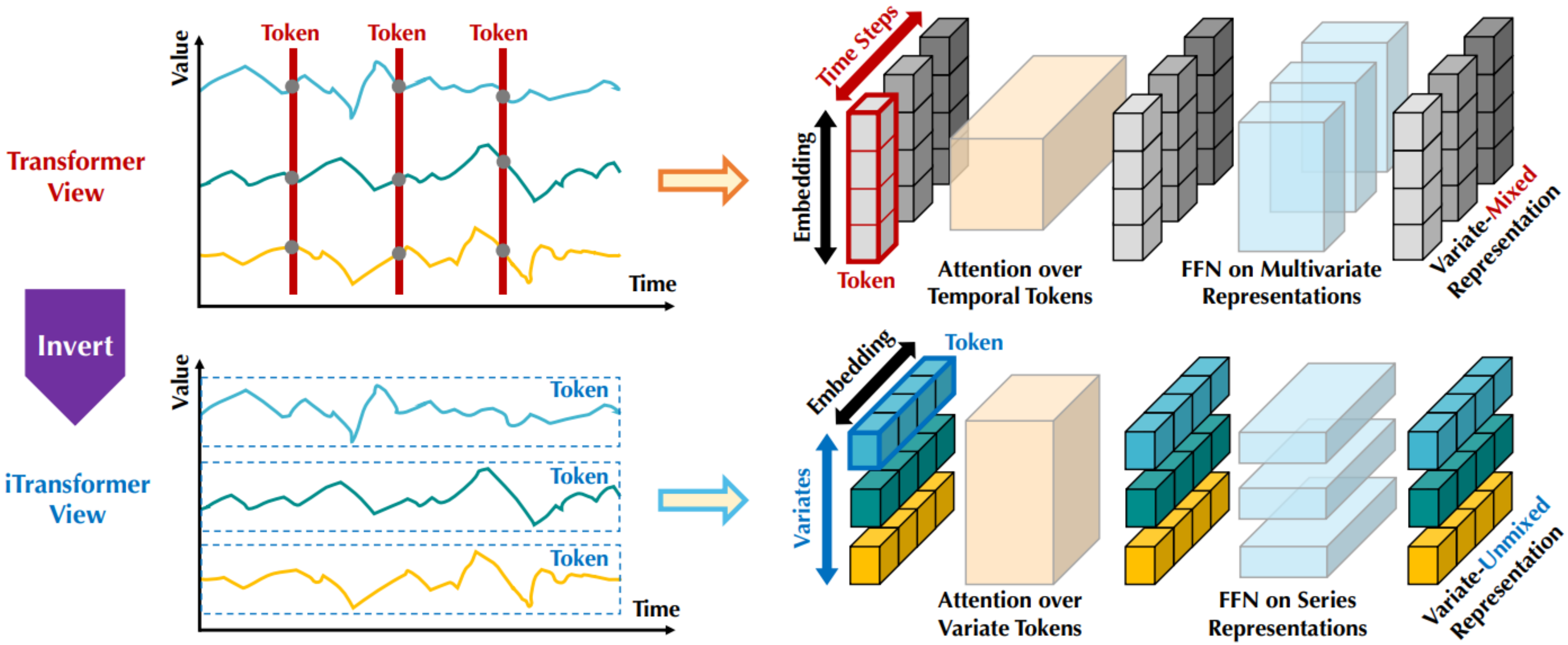
（以前是，同一时间戳的多变量做归一化）

所有序列 $\xrightarrow{\text{归一化}}$ 高斯分布

$$\mathrm{LayerNorm}(\mathbf{H})=\left\{\frac{\mathbf{h}_n-\mathrm{Mean}(\mathbf{h}_n)}{\sqrt{\mathrm{Var}(\mathbf{h}_n)}}\,\middle|\, n=1,\cdots,N\right\}$$

04

iTransformer



05

实验1：预测对比

Models	iTransformer (Ours)		RLinear (2023)		PatchTST (2023)		Crossformer (2023)		TiDE (2023)		TimesNet (2023)		DLinear (2023)		SCINet (2022a)		FEDformer (2022)		Stationary (2022b)		Autoformer (2021)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	**0.178**	**0.270**	0.219	0.298	0.216	0.304	0.244	0.334	0.251	0.344	0.192	0.295	0.212	0.300	0.268	0.365	0.214	0.327	0.193	0.296	0.227	0.338
ETT (Avg)	0.383	0.399	**0.380**	**0.392**	0.381	0.397	0.685	0.578	0.482	0.470	0.391	0.404	0.442	0.444	0.689	0.597	0.408	0.428	0.471	0.464	0.465	0.459
Exchange	0.360	**0.403**	0.378	0.417	0.367	0.404	0.940	0.707	0.370	0.413	0.416	0.443	**0.354**	0.414	0.750	0.626	0.519	0.429	0.461	0.454	0.613	0.539
Traffic	**0.428**	**0.282**	0.626	0.378	0.555	0.362	0.550	0.304	0.760	0.473	0.620	0.336	0.625	0.383	0.804	0.509	0.610	0.376	0.624	0.340	0.628	0.379
Weather	**0.258**	**0.279**	0.272	0.291	0.259	0.281	0.259	0.315	0.271	0.320	0.259	0.287	0.265	0.317	0.292	0.363	0.309	0.360	0.288	0.314	0.338	0.382
Solar-Energy	**0.233**	**0.262**	0.369	0.356	0.270	0.307	0.641	0.639	0.347	0.417	0.301	0.319	0.330	0.401	0.282	0.375	0.291	0.381	0.261	0.381	0.885	0.711
PEMS (Avg)	**0.119**	**0.218**	0.514	0.482	0.217	0.305	0.220	0.304	0.375	0.440	0.148	0.246	0.320	0.394	0.121	0.222	0.224	0.327	0.151	0.249	0.614	0.575

PEMS03/04/07/08，预测长度12/24/36/48，历史长度96

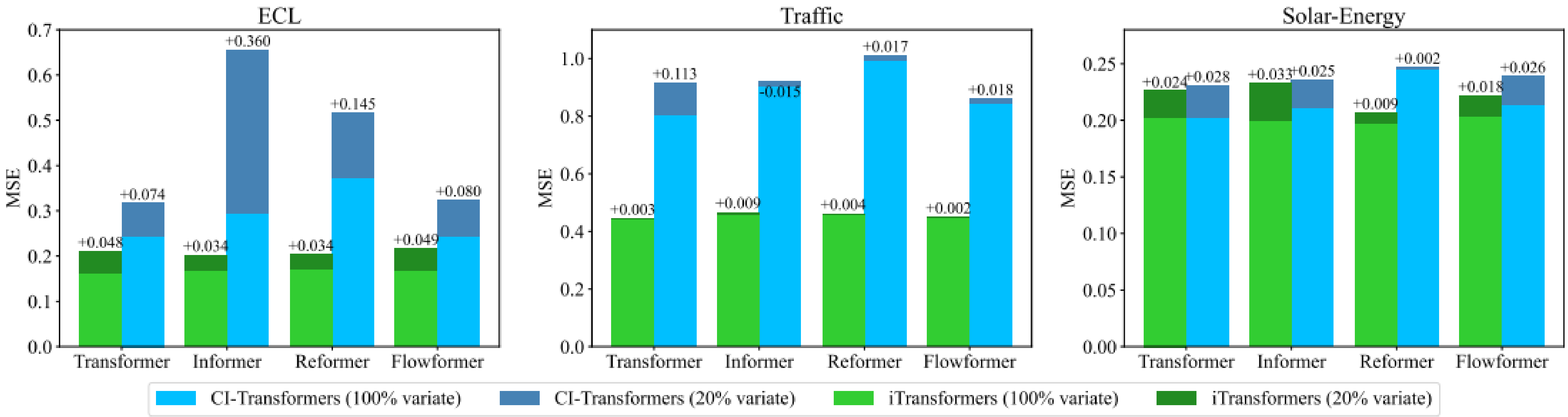
05

实验2：性能提升

Models		Transformer (2017)		Reformer (2020)		Informer (2021)		Flowformer (2022)		Flashformer (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	Original	0.277	0.372	0.338	0.422	0.311	0.397	0.267	0.359	0.285	0.377
	+Inverted	**0.178**	**0.270**	**0.208**	**0.301**	**0.216**	**0.311**	**0.210**	**0.293**	**0.206**	**0.291**
	Promotion	35.6%	27.4%	38.4%	28.7%	30.5%	21.6%	21.3%	18.6%	27.8%	22.9%
Traffic	Original	0.665	0.363	0.741	0.422	0.764	0.416	0.750	0.421	0.658	0.356
	+Inverted	**0.428**	**0.282**	**0.647**	**0.370**	**0.662**	**0.380**	**0.524**	**0.355**	**0.492**	**0.333**
	Promotion	35.6%	22.3%	12.7%	12.3%	13.3%	8.6%	30.1%	15.6%	25.2%	6.4%
Weather	Original	0.657	0.572	0.803	0.656	0.634	0.548	0.286	0.308	0.659	0.574
	+Inverted	**0.258**	**0.279**	**0.248**	**0.292**	**0.271**	**0.330**	**0.266**	**0.285**	**0.262**	**0.282**
	Promotion	60.2%	50.8%	69.2%	55.5%	57.3%	39.8%	7.2%	7.7%	60.2%	50.8%

05

实验3：泛化能力



数据： 只使用一个文件夹的20%变量训练模型

对比模型： 通道独立泛化策略

05

实验4：增加历史数据长度

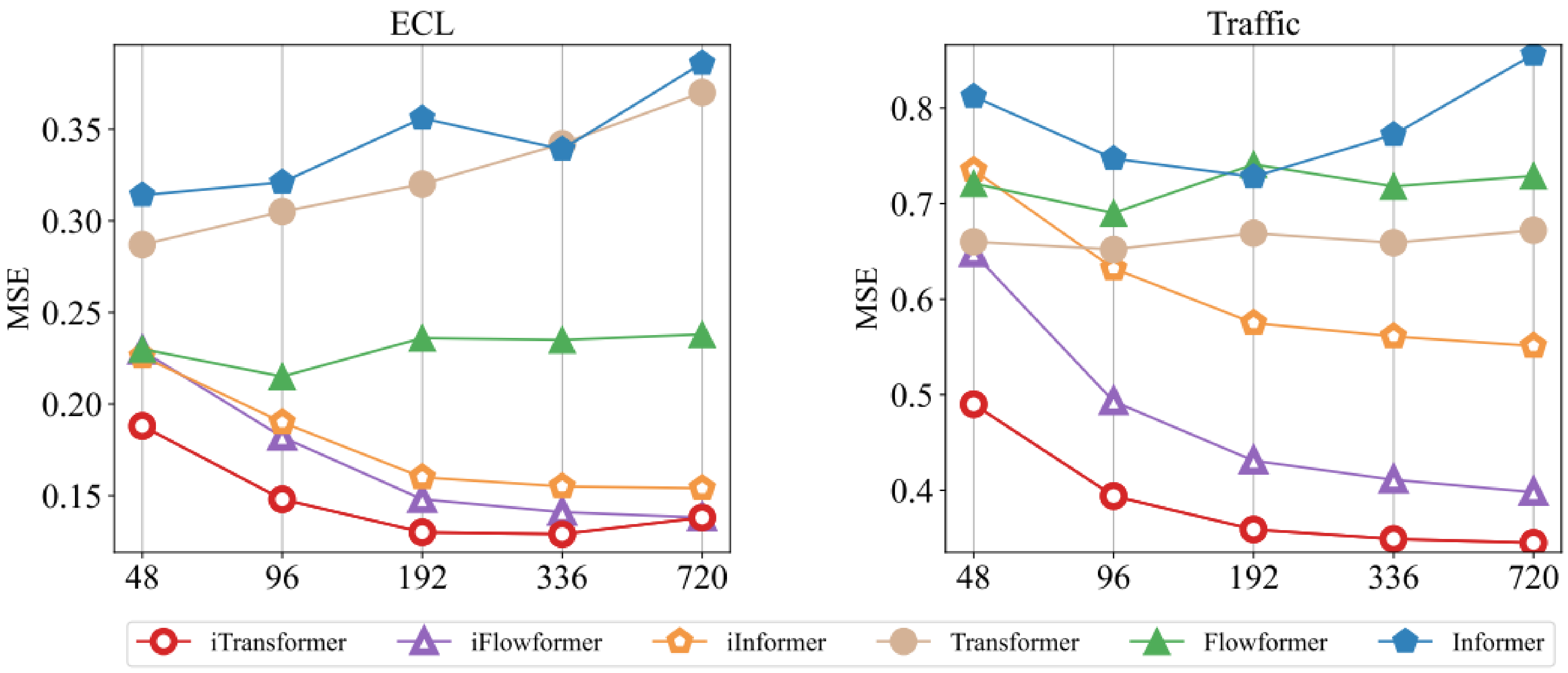


Figure 6: Forecasting performance with the lookback length $T \in \{48, 96, 192, 336, 720\}$ and fixed prediction length $S = 96$. While the performance of Transformer-based forecasters does not

输入过长：使得注意力分散，更长的历史序列对于transformer反倒效果变差

实验结果：验证了在时间维度上利用MLP的合理性

05

实验5：消融实验

Design	Variate	Temporal	ECL		Traffic		Weather		Solar-Energy	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
iTransformer	**Attention**	**FFN**	**0.178**	**0.270**	**0.428**	**0.282**	0.258	0.278	**0.233**	**0.262**
Replace	Attention	Attention	0.193	0.293	0.913	0.500	0.255	0.280	0.261	0.291
	FFN	Attention	0.202	0.300	0.863	0.499	0.258	0.283	0.285	0.317
	FFN	FFN	0.182	0.287	0.599	0.348	**0.248**	**0.274**	0.269	0.287
w/o	Attention	w/o	0.189	0.278	0.456	0.306	0.261	0.281	0.258	0.289
	w/o	FFN	0.193	0.276	0.461	0.294	0.265	0.283	0.261	0.283

06 补充

1、作者回复：

以往的Transformer（包括基于Patch的实现）将时间序列变成特征向量后，都必不可少包含一个时间维度，之后用注意力机制建模每个时序依赖，但只要时间序列的测点不是按时间对齐的，或者含有无关历史信息，这种方式建模时序依赖必然引入额外噪声，并且跨变量之间的某些Patch本身就很难具备显著的关系来帮助预测。

因此这些方法依然可能学到弱语义性的注意力图，尤其是产生非常大的计算复杂度，训练时间很长

来源：https://zhuanlan.zhihu.com/p/662250788

谢谢观看

MANY THANKS !

23.12.19