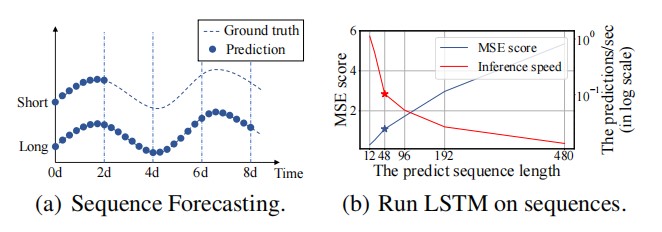
# 概要

许多实际应用需要长序列时间序列的预测，如用电量规划。长序列时间序列预测(LSTF)要求该模型具有较高的预测能力，即能够有效地捕获输出和输入之间精确的长期依赖耦合。最近的研究表明，Transformer具有提高预测能力的潜力。然而，Transformer存在几个严重的问题，阻止了它直接适用于LSTF，包括二次时间复杂度、高内存使用，以及编码器-解码器体系结构的固有限制。为了解决这些问题，我们为LSTF设计了一个高效的基于Transformer的模型，名为Informer，具有三个独特的特点：（i）一种prob稀疏自注意机制，在时间复杂度和内存使用方面实现了O(L\*logL)，在序列依赖对齐方面具有相当的性能。（ii）自我注意力提取的突出部分通过将级联层输入减半来控制注意力，并有效地处理极长的输入序列。（iii）生成式解码器虽然概念简单，但以一个正向操作预测长时间序列，而不是一步一步的方式，这大大提高了长序列预测的推理速度。在四个大规模数据集上进行的大量实验表明，Informer的性能显著优于现有的方法，并为LSTF问题提供了一种新的解决方案。

# 介绍

时间序列预测是许多领域的关键组成部分，如传感器网络监测（帕帕迪米里奥和余2006年）、能源和智能电网管理、经济和金融（朱和沙沙2002年）以及疾病传播分析(Matsubara等2014年)。在这些场景中，我们可以利用大量关于过去行为的时间序列数据来做出长期的预测，即长序列时间序列预测(LSTF)。然而，现有方法大多在短期问题设置下设计，如预测48点或以下（霍克雷特和施米杜伯1997；李等2018；于等2017；刘等2019；秦等2017；温等2017）。越来越长的序列使模型的预测能力紧张，使这一趋势阻碍了对LSTF的研究。作为实证例子，图（1）显示了真实数据集的预测结果，其中LSTM网络预测了变电站从短期（12点0.5天）到长期（480点20天）的每小时温度。当预测长度大于48点(图(1b)中的实星)，MSE性能上升到不理想，推理速度急剧下降，LSTM模型开始失效时，整体性能差距较大。



（图1）

图1：(a)LSTF可以覆盖比短序列预测更长的时间，在政策规划和投资保护方面具有至关重要的区别。(b)现有方法的预测能力限制了LSTF的性能。例如，从长度=48开始，MSE上升得高得令人无法接受，推理速度迅速下降。

LSTF面临的主要挑战是提高预测能力，以满足日益长的序列需求，这需要(a)非凡的远程比对能力和对长序列输入和输出的(b)高效操作。最近，Transformer模型在捕获长期依赖性方面表现出了比RNN模型更优越的性能。自注意机制可以将网络信号传输路径的最大长度减小到理论上最短的O（1），避免了循环结构，因此Transformer在LSTF问题上具有很大的潜力。然而，自我注意机制违反了需求(b)，因为它的L-二次计算和对L-长度的输入/输出的内存消耗。一些大规模的Transformer模型投入了资源，并在NLP任务上产生了令人印象深刻的结果(Brown等人，2020年)，但对几十个GPU的培训和昂贵的部署成本使得这些模型在现实世界的LSTF问题上负担不起。自注意机制和Transformer架构的效率成为将它们应用于LSTF问题的瓶颈。因此，在本文中，我们试图回答以下问题：我们能否改进Transformer模型，使其计算、内存和架构高效，同时保持更高的预测能力？

Vanilla Transformer(Vaswanietal.2017)在解决LSTF问题时有三个显著的局限性：

1.自注意的二次计算。自注意机制的原子操作，即规范点积，使得每层的时间复杂度和内存使用量为O(L2)。

2.长输入堆叠层的内存瓶颈。J个编码器/解码器层的堆栈使总内存使用量为O(J\*L2)，这限制了模型在接收长序列输入时的可伸缩性。

3.预测长期产出的速度急剧下降。普通Transformer的动态解码使得逐步的推理与基于RNN的模型一样慢（图(1b)）。

以往也有一些关于提高自我注意效率的工作。The Sparse Transformer（孩子等。2019），LogSparse Transformer（李等。2019），和Longformer（贝尔塔吉、彼得斯和科汉2020）都使用启发式方法来解决限制1和减少自我注意机制的复杂性到O(L\*logL)，他们的效率增益有限（邱等.2019）。Reformer(Kitaev，Kaiser和Lev斯卡娅2019)也通过局部敏感的哈希自注意实现了O(L\*logL)，但它只适用于极长的序列。最近，Linformer(Wangetal.2020)声称线性复杂度O(L)，但项目矩阵不能固定为真实世界的长序列输入，这可能有退化到O(L2)的风险。Transformer-XL(Dai等人2019)和Compressive-Transformer(Rae等人2019)使用辅助隐藏状态来捕获长期依赖关系，这可能会放大限制1，不利于打破效率瓶颈。这些工作主要集中在局限1，局限2和3在LSTF问题中仍然没有得到解决。为了提高预测能力，我们解决了所有这些限制，并在建议的信息者中实现了除效率之外的改进。

为此目的，我们的工作明确地深入探讨了这三个问题。我们研究了自注意机制的稀疏性，对网络组件进行了改进，并进行了广泛的实验。本文的贡献总结如下：

* 我们提出了成功地提高LSTF问题的预测能力，验证了类Transformer模型在捕获长序列时间序列输出和输入之间的长期依赖关系方面的潜在价值。
* 我们提出了概率稀疏自注意机制来有效地取代规范的自注意。它实现了依赖关系对齐时的O(L\*logL)时间复杂度和O(L\*logL)内存使用情况。我们提出了自注意提取操作来控制j叠加层中的注意分数，并将总空间复杂度急剧降低为*O*((2*−ε*)*L\**log*L*)，这有助于接收长序列输入。
* 我们提出了生成式解码器来获取长序列输出，只需一个向前的步骤，同时避免了在推理阶段的累积误差扩散。

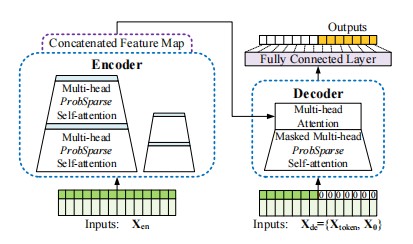


图2：Informer模型概述。左：编码器接收大量的长序列输入（绿色序列）。我们用所提出的概率稀疏自注意来代替规范的自注意。蓝色梯形是提取主导注意力的自注意提取操作，大大减小了网络的规模。该层的堆叠副本增加了鲁棒性。右：解码器接收长序列输入，将目标元素填充为零，测量特征图的加权注意组成，并立即以生成方式预测输出元素（橙色系列）。

# 准备工作

我们首先提供了LSTF问题的定义。在固定大小窗口的滚动预测设置下，我们有输入Xt={x1t，…，xLxt|xit∈Rdx}在时间t，输出是预测相应的序列Yt={y1t，…，y|yit∈Rdy}。LSTF问题鼓励比以前的工作更长的输出长度*Ly*(Cho等，2014；LSTF，维尼亚尔和Le2014)，特征维度不限于单变量情况(*dy*≥1)。

**编码解码器体系结构**：许多流行的模型被设计为将输入表示Xt“编码”为一个隐藏的状态表示Ht，并“解码”一个输出表示Yt从Ht={h1t，……，hLht}。该推理涉及一个名为“动态解码”的逐步过程，其中解码器从之前的状态hkt计算一个新的隐藏状态hk+1t和第k步的其他必要输出，然后预测(k+1)序列yt+1k。

**输入表示**:给出了一个统一的输入表示方法来增强时间序列输入的全局位置上下文和局部时间上下文。为了避免简化描述，我们将这些细节放在附录B中。

# 方法论

现有的时间序列预测方法大致可以分为两类（由于空间限制，相关工作见附录A。）。经典时间序列模型是时间序列预测的可靠工具(Box等人2015；雷1990；西格等人2017；西格、萨利纳斯和弗伦克特2016)，深度学习技术主要使用RNN及其变体开发编解码器预测范式(霍克雷特和施米休伯1997；李等2018；Yu等2017）。我们提出的Informer拥有编码器-解码器架构，同时针对LSTF问题。详见图（2），详见以下部分。

**有效的自我注意机制：**(Vaswani等人2017)中的规范自注意基于元组输入，即查询、键和值进行细化，它将比例点积执行为A(Q、K、V)=softmax(QK>/√d)V，其中Q∈RLQ×d、K∈RLK×d、V∈RLV×d和d为输入维度。为了进一步讨论自我注意机制，让qi、ki、vi分别代表Q、K、V中的第i行。按照(Tsaietal.2019)中的公式，第i个查询的注意力被定义为概率形式的核平滑器：

*A*(**q***i,* **K***,* **V**) = X *j k*(**q***i,* **k***j* ) P *l k*(**q***i,* **k***l*)**v***j* = E*p*(**k***j |***q***i*)[**v***j* ] *,* (1)

其中p(kj|qi)=k(qi，kj）/plk(qi，kl）和k(qi，kj）选择非对称指数核exp(qik>j/√d)。自我注意结合这些值，通过计算概率p(kj|qi)获得输出。它需要二次时间点积计算和O(LQLK)内存的使用，这是提高预测能力的主要缺点。

以往的一些尝试表明，自我注意概率的分布具有潜在的稀疏性，他们对所有p(kj|qi)设计了“选择性”计数策略，而不显著影响其表现。稀疏变换器(Childetal.2019)包含了行输出和列输入，其中稀疏性来自于分离的空间相关性。LogSparseTransformer(Lietal.2019)注意到自我注意的周期性模式，并迫使每个单元通过指数步长关注其前一个单元。长前者(贝尔塔吉，Peters，和Cohan2020)将之前的两个工作扩展到更复杂的稀疏配置。然而，它们仅限于从以下启发式方法进行理论分析，并采用相同的策略处理每个多头自注意，这缩小了它们的进一步改进。

为了激发我们的方法，我们首先对典型自我注意的习得注意模式进行定性评估。“稀疏性”自我注意得分形成了一个长尾分布(详见附录C)，也就是说，一些点积对导致了主要的注意，而另一些则产生了微不足道的注意。那么，下一个问题是如何区分它们呢？

从等式中查询稀疏度度量（1），第i个查询对所有键的关注被定义为一个概率p(kj|qi)，输出是它与值v的组成。主要的点积对鼓励相应的查询的注意概率分布远离均匀分布。如果p(kj|qi)接近于均匀分布q(kj|qi)=1/LK，自注意就变成值V的平凡和，对住宅输入是冗余的。自然地，分布p和q之间的“相似性”可以用来区分“重要的”查询。我们通过库贝克-莱伯勒散度KL(q||p)=lnPLKL=1eqik>/√d−L1KPLKj=1qik>j/√d−lnLK来测量“相似性”。下降常数，我们定义我查询的稀疏性测量M(气，K)=lnLKXj=1eqik>j√d−1LKLKXj=1qik>j√d，（2）第一项Log-Sum-Exp(LSE)气的关键，第二项是算术平均值。如果第i个查询获得较大的M(qi，K)，它的注意概率p更加“多样化”，并且在长尾自注意分布的头域中包含主要的点积对。

基于提出的测量，我们有的自我关注允许每个键只参加u主导查询：(Q，K，V)=Softmax(QK>√)V，（3）问是一个稀疏的相同大小的矩阵，它只包含Top-u查询在稀疏性测量M(q，K)。在恒定采样因子=的控制下，我们设置u=·lnLQ，这使得=只需要为每个查询键查找计算O(lnLQ)点积，层内存使用保持O(LKlnLQ)。在多头视角下，这种注意力为每个头部生成不同的稀疏查询键对，从而避免了严重的信息损失。

然而，遍历测量M(qi，K)的所有查询需要计算每个点积对，即二次O(LQLK)，除了LSE操作存在潜在的数值稳定性问题。在此基础上，我们提出了一个经验近似的有效获取查询稀疏度测量。

引理1。对于键集K中的每个查询qi∈Rd和kj∈Rd，我们有绑定为lnLK≤M(qi，K)≤maxj{qik>j/√d}−L1KPLKj=1{qik>j/√d}+lnLK。当气∈K时，它也成立。

从引理1(证明见附录D.1)，我们提出最大平均测量为M(qi，K)=maxj{qik>j√d}−1LKLKXj=1qik>j√d . (4)

在命题1的边界松弛过程中，顶u的范围近似成立(见附录D.2)。在长尾分布下，我们只需要随机抽样U=LKlnLQ点积对来计算M(qi，K)，即将其他对填充为零。然后，我们从它们中选择稀疏的Top-u作为q。M(qi，K)中的最大算子对零值的敏感性较低，并且是数值稳定的。在实际应用中，查询和键的输入长度在自注意计算中通常是相等效的，即LQ=LK=L，使得总的自注意时间复杂度和空间复杂度为O(LlnL)。

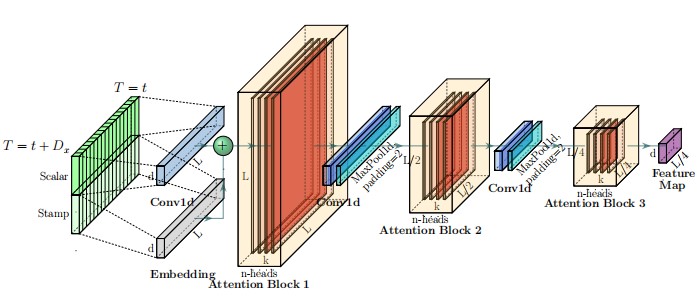


图3：Informer的编码器中的单个堆栈。（1）水平堆栈代表图（2）中编码器的一个编码器副本。（2）所给出的一个是接收整个输入序列的主堆栈。然后第二个堆栈取输入的半片，然后随后的堆栈重复。（3）红色的层是点积矩阵，通过在每一层上应用自注意提取，它们得到级联减少。（4）将所有堆栈的特性映射连接为编码器的输出。

编码器：允许在内存使用限制下处理更长的顺序输入。编码器被设计用来提取长序列输入的鲁棒的长期依赖关系。

在输入表示之后，第t个序列输入Xt被塑造成一个矩阵Xten∈RLx×模型。为了清晰起见，我们在图（3）中给出了一个编码器的草图。

自注意提取是概率自注意机制的自然结果，编码器的特征图具有冗余组合。我们利用提取操作对具有优势特征的优越图进行特权，并在下一层制作聚焦自注意特征图。它急剧地修剪了输入的时间维度，可以看到图（3）中注意块的n个头权重矩阵（重叠的红色方块）。受扩张卷积的启发(Yu、科尔敦和芬克豪瑟2017；Gupta和Rush2017)，我们的“提取”过程从第j层向前延伸到(第j+1层)，如下：

**X***tj*+1 = MaxPool ELU( Conv1d([**X***tj* ]AB) ) *,* (5)

其中，AB表示注意力块。它包含了多头概率稀疏自注意和基本操作，其中Conv1d（·）在时间维上使用ELU（·）激活函数执行一维卷积滤波器和=2016)。我们添加了一个步幅为2的最大池化层，并在堆叠一个层后将Xt下采样到它的半层中，这将整个内存使用减少为O（（2）LlogL），其中是一个很小的数字。为了增强蒸馏操作的鲁棒性，我们构建了将输入减半的主堆栈副本，并通过每次减少一层来逐步减少自注意蒸馏层的数量，就像图（2）中的金字塔一样，这样它们的输出维度就对齐。因此，我们连接了所有堆栈的输出，并得到了编码器的最终隐藏表示。

我们使用图（2）中的一个标准解码器结构(Vaswanietal.2017)，它由两个相同的多头注意层的堆栈组成。然而，生成推理被用来缓解长时间预测中的速度下降。我们给解码器与以下向量Xtde=Concat(Xt标记，Xt0)∈R（令牌+李）×模型，（6）Xt标记∈标记×模型是开始标记，Xt0∈RLy×模型是一个目标序列的占位符（设置标量为0）。通过将掩模点产品设置为−∞，将掩模多头注意应用于概率稀疏自注意计算中。它防止每个位置注意接下来的位置，从而避免了自动回归。全连接层获得最终输出，其超大尺寸dy取决于我们执行的是单变量预测还是多变量预测。

生成推理开始令牌被有效地应用于NLP的“动态解码”(Devlinetal.2018)，我们将其扩展为生成的方式。我们没有选择特定的标记作为令牌，而是在输入序列中采样一个令牌长序列，例如输出序列之前的早期切片。以预测168点为例（实验部分为7天温度预测），我们将目标序列前已知的5天作为“星令牌”，用Xde={X5d，X0}输入生成式推理解码器。x0包含目标序列的时间戳，即在目标周上的上下文。然后，我们提出的解码器通过一个正向过程来预测输出，而不是在传统的编解码器架构中进行耗时的“动态解码”。在计算效率部分给出了详细的性能比较。

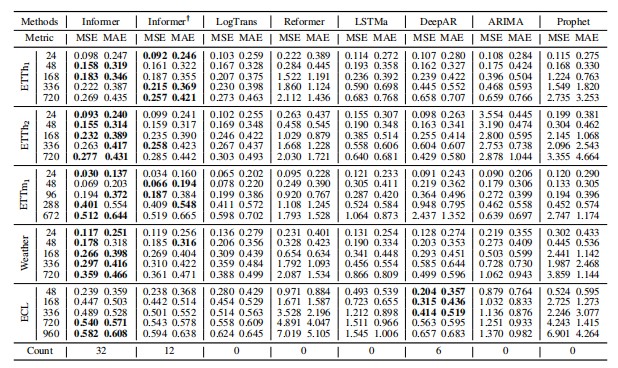
损失函数我们在目标序列的预测w.r.t上选择MSE损失函数，然后将损失从解码器的输出传播回整个模型。

表1：4个数据集（5个情况）的单变量长序列时间序列预测结果

# 实验

**数据集**

我们在四个数据集上进行了广泛的实验，包括2个收集的LSTF真实数据集和2个公共基准数据集。

ETT（电力Transformer温度）2：ETT是电力长期部署的重要指标。我们收集了来自中国两个独立的县的2年数据。为了探索LSTF问题的粒度，我们创建了单独的数据集，如1小时级的{ETTh1，ETTh2}，15分钟级的ETTm1。每个数据点由目标值“油温”和6个功率负荷特征组成。列车测试是12月4个月。

ECL（用电负荷）3：收集321个客户的用电量(Kwh)。由于数据缺失(Lietal.2019)，我们将数据集转换为2年的每小时消耗量，并将“MT320”设置为目标值。列车测试是15个月。

天气4：该数据集包含了近1600个美国地点的当地气候数据，从2010年到2013年的四年的数据，其中的数据点每1小时收集一次。每个数据点包括目标值“湿球”和11个气候特征。列车测试为10月10月28日。

**实验细节**

我们简要地总结了基本知识，并在附录E中给出了更多关于网络组件和设置的信息。

**基线**：我们选择了五种时间序列预测方法进行比较，包括ARIMA(Ariyo、阿德wumi和阿约2014年)、先知（泰勒和莱瑟姆2018年）、LSTMa（巴赫达瑙、乔和本吉奥2015年)、LSTnet(Lai等人2018年)和DeepAR（弗伦克特、萨利纳斯和加索斯2017年）。为了更好地探索概率稀疏自注意在我们提出的Informer中的表现，我们在实验中加入了典型的自注意变体（Informer）、有效的变体改革者（凯塔耶夫、凯泽和列夫斯卡娅2019）和最相关的工作逻辑稀疏自注意(Li等人2019)。网络组件的详细信息见附录E.1。

**超参数调优**：我们对超参数进行网格搜索，详细的范围在附录E.3中给出。输入者在编码器中包含一个3层堆栈和一个1层堆栈（1/4个输入），以及一个2层解码器。我们提出的方法采用Adam优化器进行优化，其学习速率从1e−4开始，每个历元衰减小2倍。时代总数为8个，并有适当的提前停止。我们设置了推荐的比较方法，批量大小为32。设置：每个数据集的输入都是零均值归一化的。

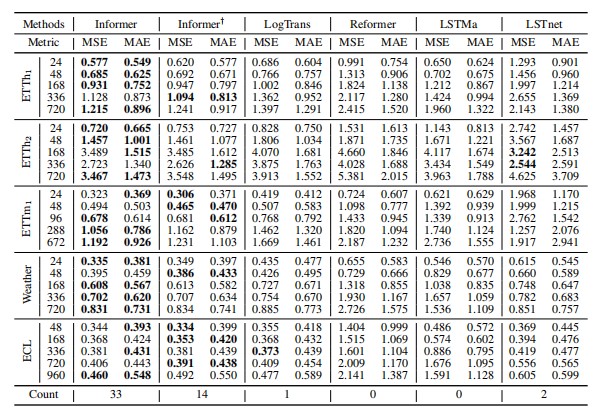


表2：4个数据集（5例）的多元长序列时间序列预测结果。

在LSTF设置下，我们在ETTm、ECL、天气、6h、12d、2d、2d、7d、14d、30d、168h。指标：我们使用两个评估指标，包括MSE=1nPni=1(y−ˆy)2和MAE=1nPni=1|y−ˆy|（多变量预测的平均），并使用步幅=1滚动整个集合。平台：所有型号都在一个NvidiaV100 32GBGPU上进行训练/测试。源代码可以在https://github.com/zhouhaoyi/Informer2020上获得。

**结果与分析**

表1和表2总结了所有方法在4个数据集上的单变量/多变量评价结果。作为预测能力的更高要求，我们逐渐延长了预测范围，其中LSTF问题设置被精确控制为每种方法在一个GPU上处理。最好的结果用粗体字突出显示。单变量时间序列预测在此设置下，每种方法都可以作为一个单一变量进行时间序列的预测。从表1，我们可以观察到：（1）提出模型Informer显著提高推理性能（获胜计数在最后一列）在所有数据集，和他们的预测误差平稳上升缓慢预测水平增长，这证明了Informer的成功提高LSTF的预测能力问题。（2）Informer主要在获胜计数方面击败了其规范的退化Informer，即32>12，它支持查询稀疏性假设，提供一个可比的注意特征图。我们提出的方法也优于最相关的工作日志转换和变形器。我们注意到，改革器在LSTF中保持了动态解码，并且在LSTF中表现较差，而其他方法则受益于生成式解码器作为非自回归预测器。（3）Informer模型的结果明显优于递归神经网络LSTMa。我们的方法的MSE分别下降了26.8%（168）、52.4%（336）和60.1%（720）。这揭示了自衰变中较短的网络路径。

多变量时间序列预测在此设置中，一些单变量方法是不合适的，而LSTnet是最先进的基线。相反，我们提出的Informer很容易通过调整最终的FCN层而从单变量预测转变为多变量预测。从表2中，我们观察到：（1）所提出的模型Informer大大优于其他方法，在单变量设置中的发现1和2仍然适用于多变量时间序列。（2）Informer模型比基于RNN的LSTMa和基于cnn的LSTnet模型的结果更好，MSE平均下降了26.6%（168）、28.2%（336）、34.3%（720）。与单变量的结果相比，压倒性的性能降低了，这种现象可能是由特征维数的预测能力的各向异性引起的。这超出了本文的研究范围，我们将在今后的工作中进行探讨。

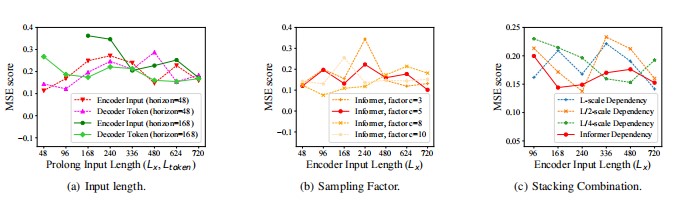


图4：Informer中三个分量的参数灵敏度。

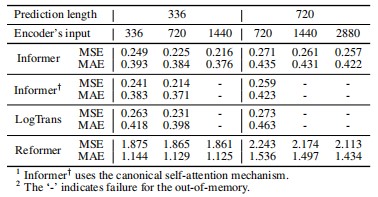


表3：概率稀自我注意机制的消融研究。

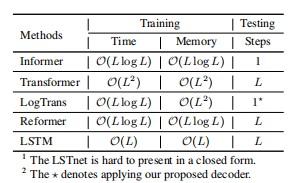


表4：每层l相关计算静态。

具有粒度考虑的LSTF我们执行了一个额外的比较来探索具有各种粒度的性能。ETTm1（分钟级）的序列{96, 288, 672}与ETTh1（小时级）的序列{24, 48, 168}对齐。即使序列处于不同的粒度级别，消息者执行者的性能也优于其他基线。

**参数灵敏度**

我们在单变量设置下对所提出的ETTh1模型进行了敏感性分析。输入长度：在图(4a)中，当预测短序列（如48）时，最初增加编码器/解码器的输入长度会降低性能，但进一步增加会导致MSE下降，因为它带来了重复的短期模式。然而，在预测长序列时输入时间越长，MSE就越低（如168）。因为较长的编码器输入可能包含更多的依赖关系，而较长的解码器令牌具有丰富的本地信息。抽样因子：抽样因子控制了Eq.（3）中概率疏自注意的信息带宽。我们从小因子（=3）开始，到大因子，一般性能略有提高，最后在图(4b)中稳定下来。验证了我们在自注意机制中存在冗余点产物对的查询稀疏性假设。我们在实践中设置了样本因子c=5（红线）。层堆叠的组合：层的复制品对于自注意提取是互补的，我们在图(4c)中研究了每个堆叠{L、L/2、L/4}的行为。堆栈越长，s就越多

消融研究：Informer的工作效果如何？

我们还在ETTh1上进行了一些考虑到消融作用的额外实验。

在总体结果表1和表2中，我们限制了问题的设置，以使规范自注意的内存使用可行。在本研究中，我们将我们的方法与LogTrans和改革者进行了比较，并彻底探讨了它们的极端性能。为了隔离内存效率问题，我们首先将设置减少为{批处理大小=8，heads=8，dim=64}，并在单变量情况下维护其他设置。在表3中，概率稀疏的自注意表现出优于对应的表现。LogTrans在极端情况下会得到OOM，因为它的公共实现是完全关注的掩码，它仍然有O(L2)内存使用。我们提出的问题稀疏自注意避免了等式中查询稀疏假设带来的简单性带来的问题（4），参考了附录E.2中的伪代码，并达到了较小的内存使用量。

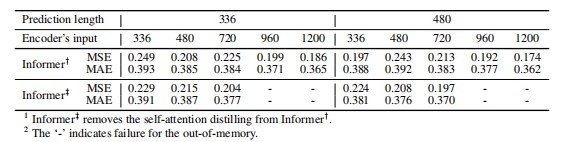


表5：自我注意力提取的消融研究

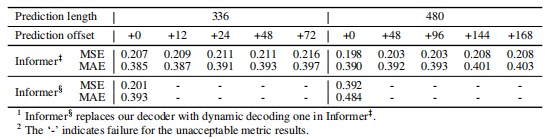


表6：生成式解码器的消融研究

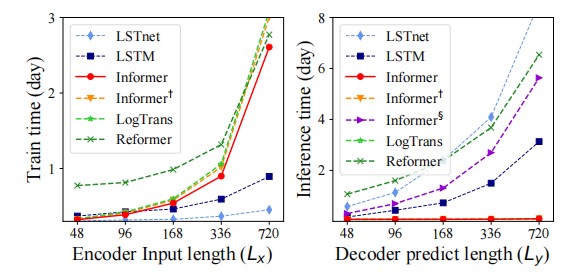


图5：训练/测试阶段的总运行时间。

在本研究中，我们以“Informer”为基准，消除了概率稀疏自注意的附加影响。另一个实验设置与单变量时间序列的设置相一致。从表5可以看出，Informer完成了所有的实验，利用长序列输入获得了更好的性能。比较方法Informer去除蒸馏操作，并以较长的输入达到OOM（>720）。关于长序列输入在LSTF问题中的好处，我们得出结论，自我注意提取是值得采用的，特别是当需要更长时间的预测时。

在生成式解码器的性能中，我们证明了我们的解码器在获取“生成”结果方面的潜在价值。与现有的方法不同，标签和输出在训练和推理中被迫对齐，我们提出的解码器的预测仅依赖于时间戳，它可以用偏移量来预测。从表6可以看出，告者的一般预测性能随着偏移量的增加而抵抗，而动态预测译码则失效。它证明了解码器捕获任意输出之间的单个长期依赖关系和避免错误积累的能力。

**计算效率**

对于多变量设置和所有方法当前最好的实现，我们在图（5）中进行了严格的运行时比较。在训练阶段，Informer（红线）在基于Transformer的方法中达到了最好的训练效率。在测试阶段，我们的方法比其他使用生成式解码的方法要快得多。表4总结了理论时间复杂度和内存使用情况的比较。消息论者的性能与运行时实验结果一致。请注意，LogTrans专注于改进自注意机制，并且我们应用我们提出的LogTrans解码器来进行公平的比较(即？在表4中)。

# 结论

本文研究了长序列时间序列预测问题，并提出了长序列预测问题。具体地说，我们设计了概率稀疏自注意机制和提取操作来处理普通Transformer中的二次时间复杂度和二次内存使用的挑战。此外，精心设计的生成解码器减轻了传统编解码器结构的局限性。在真实数据上的实验证明了Informer对提高LSTF问题预测能力的有效性。

# 附录

### 附录A 相关工作

下面，我们将提供一个关于长序列时间序列预测(LSTF)问题的文献综述。

时间序列预测现有的时间序列预测方法大致可以分为两类：经典模型和基于深度学习的方法。经典的时间序列模型是时间序列预测的可靠工作工具，具有可解释性和理论保证等诱人的特性(Boxetal.2015；Ray1990)。现代扩展包括对缺失数据的支持(Seeger等人，2017年)和多种数据类型(Seeger、Salinas、萨利纳斯和Flunkert，2016年)。基于深度学习的方法主要利用RNN及其变体开发序列到序列的预测范式，实现了开创性的性能(霍克雷特和施米德胡伯，1997；李等，2018；Yu等，2017)。尽管取得了实质性的进展，现有算法仍无法预测长序列时间序列。典型的最先进的方法(Seeger等2017；西格、萨利纳斯和弗伦克特2016)，特别是深度学习方法(Yu等2017；秦等2017；弗伦克特、萨利纳斯和加索斯2017；穆克吉等人2018；温等人2017)，仍然是一个序列预测范式。

长序列输入问题从上面的讨论中，我们提到了关于长序列时间序列输入(LSTI)问题的第二个限制。我们将探讨相关的工作，并对我们的LSTF问题进行比较。研究人员在实践中截断/总结/采样输入序列，以处理一个非常长的序列，但在做出准确的预测时，可能会丢失有价值的数据。截断的BPTT(Aicher，Foti，和Fox，2019年)没有修改输入，而是只使用最后的时间步长来估计权重更新中的梯度，而辅助损失(Trinh等人，2018年)通过添加辅助梯度来增强梯度流量。其他尝试包括循环高速公路网络(Zillyetal.2017)和引导规则化器(Cao和Xu2019)。这些方法试图改进递归网络长路径中的梯度流，但随着LSTI问题中序列长度的增加，其性能受到了限制。基于cnn的方法(Stoller等人，2019；Bai，Kolter和2018)使用卷积滤波器捕获长期依赖，它们的接受域随着层的堆积呈指数级增长，这损害了序列对齐。在LSTI问题中，主要任务是提高模型接收长序列输入的能力，并从这些输入中提取长期依赖性。但LSTF问题旨在提高模型对长序列输出的预测能力，这需要建立输出和输入之间的长期依赖关系。因此，上述方法不能直接用于LSTF。

注意模型Bahdanau等人首先提出了成瘾注意(Bahdanau、Cho和Bengio2015)，以改进翻译任务中编解码器结构的单词对齐。然后，它的变体(龙、Pham和曼宁2015)提出了广泛使用的位置、一般和点产品的关注。流行的基于自关注的Transformer(Vaswanietal.2017)最近被提出作为序列建模的新思维，并取得了巨大的成功，特别是在NLP领域。通过将其应用于翻译、语音、音乐和图像生成，验证了更好的序列对齐的能力。在我们的工作中，Informer利用了其序列比对的能力，使其适应LSTF问题。

基于Transformer的时间序列模型最相关的工作(Song等人2018；Ma等人2019；李等人2019)都从在时间序列数据中应用Transformer的轨迹开始，在使用香草Transformer时LSTF预测失败。以及其他一些作品(Child等人2019；Li等人2019)注意到自我注意机制的稀疏性，我们已经在主要背景下进行了讨论。

### 附录B：统一的输入表示法

RNN模型(舒斯特尔和帕利瓦尔1997年；霍克雷特和施米德胡伯1997年；钟等人2014；苏斯克弗，维尼亚尔斯和勒2014；秦等人2017；Chang等人2018)通过循环结构本身捕捉时间序列模式，几乎不依赖时间戳。香草Transformer（Vaswani等人，2017；Devlin等人，2018）使用点状自我注意机制，时间戳作为局部位置背景。然而，在LSTF问题中，捕获长期独立的能力需要全局信息，如分层的时间戳（周、月和年）和不可知论的时间戳（假期、事件）。这些数据很难在规范的自注意中得到利用，因此编码器和解码器之间的查询键不匹配会导致预测性能的潜在退化。我们提出了一个统一的输入表示来缓解这个问题，图（6）给出了一个直观的概述。

假设我们有第t个序列输入，X、t和p个类型的全局时间戳，输入表示后的特征维数是d模型。我们首先使用固定位置嵌入来保留局部上下文：

PE(pos,2j) = sin(pos/(2Lx)2j/dmodel )

PE(pos,2j+1) = cos(pos/(2Lx)2j/dmodel )

其中，j∈{1，……，bdmodel/2c}。每个全局时间戳都被一个可学习的戳嵌入SE(pos)使用，其声音大小有限（高达60，即以分钟作为最小粒度）。也就是说，自注意的相似性计算可以访问全局上下文，并且计算消耗对长输入是负担得起的。为了对齐维数，我们将标量上下文xti投影到具有一维卷积滤波器（核宽度=3，步幅=1）的d模型-暗向量uti中。因此，我们有喂养向量Xt供给[i]=αuti+PE(Lx×(t−1)+i，)+Xp[SE(Lx×(t−1)+i)]p，（8），其中i∈{1，…，Lx}，和α是平衡标量投影和局部/全局嵌入之间的大小的因素。如果序列输入已经被规范化，我们建议使用α=1。

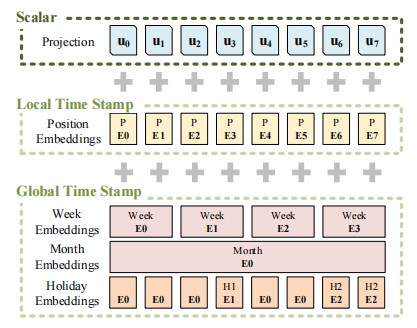


图6：Informer的输入表示形式。输入的嵌入由三个独立的部分组成，一个标量投影、局部时间戳（位置）和全局时间戳嵌入（分钟、小时、周、月、假日等）。

### 附录C 自注意特征图中的长尾分布

我们在ETTh1数据集上进行了香草变换，以研究自注意特征图的分布。我们选择{Head1，Head7}@层1的注意力分数。图（7）中的蓝线形成了一个长尾分布，即一些点积对贡献了主要的关注，其他的可以忽略。

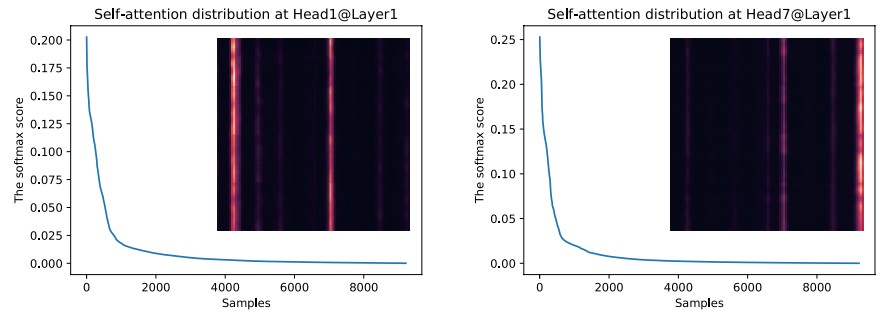


图7：在ETTh1数据集上训练的4层规范Transformer的自我注意方面的Softmax得分。

### 附录D 证明的细节

Lemma1的证明

证明。对于个体气，我们可以将离散键放宽为连续的d维变量，即向量kj。查询稀疏度度量被定义为M(qi，K)=lnPLKj=1eqik>j/√d−L1KPLKj=1(qik>j/√d)。首先，我们来看看不平等的左边部分。对于每个查询qi，M(qi，K)的第一项成为一个固定查询qi和所有键的内积的对数和exp，我们可以定义fi(K)=lnPLKj=1eqik>j/√d。从等式中开始（2）在对数和exp网络（卡拉菲尔，戈伯特和波塞里2018），进一步分析，函数fi(K)是凸的。此外，fi(K)添加一个kj的线性组合，使M(qi，K)成为一个固定查询的凸函数。然后我们可以推导关于个别向量kj为∂M(气，K)∂kj=eqik>j/√PLKj=1eqik>j/√·气√−L1√·气√。为了达到最小值，我们让~∇M(qi)=~0，以下条件为qik>1+lnLK=···=qik>j+lnLK=···=lnPLKj=1eqik>j。当然，它需要k1=k2=···=kLK，我们有测量的最小值为lnLK，即

M(qi, K) ≥ lnLK

其次，我们研究了不平等的正确部分。如果我们选择最大的内积maxj{qik>/√d}，很容易M(气，K)=LKXj=1e问√>j−1LKXj=1(问>√d)≤ln(LK·马克斯{问>√})−1LKLKXj=1（问>√）=LNLK+{问>√}−1LKLKXj=1(问>j√d)。（10）结合了等式（14）和等式（15），我们有引理1的结果。当键集与查询集相同时，上面的讨论也成立。