神经先知。可解释的规模化预测

Oskar Triebe[a,1,*], Hansika Hewamalage[c], Polina Pilyugina[d], Nikolay Laptev[b], Christoph Bergmeir[c], Ram Rajagopal[a]

[a]斯坦福大学[b] Facebook, Inc.[c]莫纳什大学[d] Skoltech大学

摘要

我们介绍NeuralProphet，它是Facebook Prophet的后继者，为可解释的、可扩展的和用户友好的预测框架设立了一个行业标准。随着时间序列数据的激增，可解释的预测仍然是商业和运营决策的一项挑战任务。我们需要混合解决方案来弥补可解释的经典方法和可扩展的深度学习模型之间的差距。我们认为先知是这种解决方案的先导。然而，Prophet缺乏本地背景，而这对于预测近期的未来是至关重要的，并且由于其斯坦的后端，扩展起来很有挑战性。

NeuralProphet是一个基于PyTorch的混合预测框架，用标准的深度学习方法进行训练，使开发者很容易扩展该框架。本地环境是通过自动回归和协变量模块引入的，这些模块可以配置为经典的线性回归或神经网络。除此之外，NeuralProphet还保留了Prophet的设计理念，并提供相同的基本模型组件。

我们的结果表明，在一组生成的时间序列上，NeuralProphet产生了与Prophet质量相当或更高的可解释性预测成分。NeuralProphet在不同的真实世界数据集上的表现优于Prophet。对于短期到中期的预测，NeuralProphet将预测准确率提高了55%到92%。

关键词 可解释，预测，神经网络，时间序列，深度学习

*通讯作者。*电子邮件地址：*triebe@stanford.edu
[1]Github资源库的维护者。

1. 简介

1.1. 背景介绍

时间序列数据在大多数工业部门都很突出。虽然在经济学等理论和应用中得到了广泛的研究，但工业应用中的实际预测直到最近才得到广泛的关注。

由于许多公司已经大大改善了他们的数据收集，以至于现在的数据可用性超过了他们的数据分析能力，如何处理和预测工业时间序列正迅速成为一个重要的话题。通常情况下，负责解决某一特定问题的人不是时间序列专家，而是一个从业预测者。

准确的时间序列预测对决策过程至关重要，特别是在基础设施规划、预算分配和供应链管理方面。过度预测和预测不足都会造成很大的损失。在应用中，预测为商业或运营决策提供信息，并可能产生致命的后果，这通常要求模型是可解释的。一个常见的方法是将预测分解成可单独解释的部分。这允许领域专家对预测进行审查，并在适当的时候进行调整，这种程序被称为 "人在环"。

经典的时间序列方法。
预测领域在传统上是由统计技术主导的。这在许多早期的预测比赛中最为明显，如NN3、NN5和M3（Makridakis & Hibon, 2000; Crone, 2008）。经典模型，如自回归综合移动平均模型（ARIMA）和指数平滑模型（ETS）已被充分研究，并提供了可解释的成分。然而，它们的限制性假设和参数化性质限制了它们在现实世界中的应用表现。熟练的预测专家可以转换数据和组合算法，以满足特定条件，获得更好的性能。这需要在应用本身和经典的时间序列建模方面有深厚的领域知识。

机器学习方法。
基于机器学习（ML）的模型在早期的预测比赛中一直表现不佳。神经网络（NN）甚至一度被认为在预测方面没有竞争力（Hyndman,
2018）。它们在文献中被进一步批评为黑箱性质，如Makridakis, Spiliotis & Assimakopoulos（2018）。然而，随着大规模时间序列可用性的爆炸，基于NN的数据驱动技术重新获得了

他们在预测方面的普及。可用的数据量已经不足以让ML和深度学习（DL）技术进行良好的训练。然而，这些模型的可解释性仍然是预测领域的一个公开研究问题。此外，它们往往需要大量的工程努力来预处理数据和微调超参数。

因此，大多数参与不同行业的非专业预测从业者不会为他们的特定任务使用最准确的最先进的模型。相反，他们更感兴趣的是找到一个合理准确的模型，但要符合可解释性、可扩展性和最小的调整。这些通常被看作是工业中预测应用的要求。

实践者倾向于选择传统的统计技术，尽管它们的预测性能很差，但在功能上是方便用户和可解释的。因此，需要开发新的方法，以弥补经典时间序列建模和基于ML的方法之间的差距。

混合方法。 混合方法的先驱，Facebook Prophet（Taylor & Letham, 2017），提供了一个可解释的模型，可扩展到许多预测应用。该预测框架为新手提供了完全自动化，为领域专家提供了微调能力。Prophet仍然是数据科学家最先使用的预测包之一，也是他们随着技能增长经常继续使用的少数预测包之一。它使经典的时间序列预测变得平易近人，对广泛的人群有用。

然而，它在关键特征方面的局限性，如缺乏本地环境和可扩展性，给用户带来了挑战。缺乏本地环境，而本地环境对于预测近期的未来至关重要，这限制了先知在工业应用中的实用性。因为先知是建立在斯坦（Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li & Riddell, 2017）这种概率编程语言之上的，所以很难扩展原始预测库。

*我们的贡献。*受到Prophet对预测社区影响的启发，我们的目标是继续推进预测工具的民主化，使混合模型像Prophet的经典模型一样容易使用。作为向真正的混合方法迈出的一步，我们介绍了NeuralProphet，一个建立在PyTorch上的用户友好和可解释的预测工具（Paszke, Gross, Massa, Lerer, Bradbury, Chanan, Killeen, Lin, Gimelshein, Antiga, Desmaison, Kopf, Yang, DeVito, Raison, Tejani, Chilamkurthy, Steiner, Fang, Bai & Chintala, 2019）。NeuralProphet融合了经典的

Prophet软件包引入的时间序列组件与NN模块组成一个混合模型，使其能够适应非线性动态。两个这样的NN组件是自动回归和协变量模块。通过使用PyTorch作为后端，NeuralProphet可以随着深度学习的最新创新而更新。我们通过预选强大的默认值和自动化许多建模决策，为非专业人士提供了一个方便的预测工具。高级用户可能也会发现这个软件包很有用，因为他们可以通过许多自定义选项纳入领域知识。总的来说，NeuralProphet抽象了很多预测领域的知识，并结合了ML的最佳实践，所以用户可以专注于手头的工作。

在 本 文 中 。

我们详细介绍了NeuralProphet模型及其组成部分，并将其与Prophet进行比较。方法部分描述了实验设置和评估。合成数据结果量化了分解的可解释预测组件的准确性。现实世界的基准显示了在一组不同的单变量时间序列上现实的开箱即用的预测性能。最后，我们对研究结果进行了总结，并就根据手头的任务使用哪种模型提供了建议。

2. 神经先知模型

2.1. 模型组件

NeuralProphet模型的一个核心概念是它的模块组合性。该模型由各个模块组成，每个模块都为预测提供一个加法成分。大多数组件也可以被配置为由趋势缩放的乘法效应。每个模块都有各自的输入和建模过程。然而，所有模块都必须产生h个输出，其中h个定义了一次性预测未来的步骤数。这些加起来就是时间序列未来值$\hat{y}_t$, ..., $\hat{y}_{t+h-1}$ 的预测值y_t , ..., y_{t+h-1} 。如果模型只是时间依赖性的，可以产生任意数量的预测值。在下面的描述中，这种特殊情况在数学上将被处理为等同于h=1的一步超前预测。

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t) \quad (1)$$

其中。

$T(t)$ = 时间t的趋势

$S(t)$ = 时间t的季节性影响

$E(t)$ = 时间t的事件和假日效应

$F(t)$ = 未来已知外生变量在时间t的回归效应

$A(t)$ = 基于过去观察结果的时间t的自动回归效应

$L(t)$ = 外生变量的滞后观测值在时间t的回归效应

所有的模型组成模块都可以单独配置并组合成模型。如果所有模块都关闭，只有一个静态偏移参数被拟合为趋势成分。默认情况下，只有趋势和季节性模块被激活。完整的模型总结为公式1

在以下几个小节中，我们将更详细地讨论每个组成部分。

2.1.1. 趋势

建立趋势模型的一个经典方法是将其作为偏移量m和增长率k的组合。在某一时间t_1，用增长率乘以自起点t_0在偏移量m之上的时间差，就可以得到趋势效应。

$$T(t_1) = T(t_0) + k \cdot \Delta_t = m + k \cdot (t_1 - t_0) \quad (2)$$

NeuralProphet用这种经典方法对趋势部分进行建模，但允许增长率在一些地方发生变化。因此，趋势被建模为一个连续的片状线性序列。这导致了一种可解释的，但非线性形式的趋势建模。解释起来很简单，因为在两点之间的一段，趋势效应是由稳定的增长率乘以时间的差异而得到的。我们可以通过定义一个随时间变化的增长率$\delta(t)$和一个随时间变化的偏移量$\rho(t)$来概括该趋势。

$$T(t) = \delta(t) \cdot t + \rho(t)$$

片状线性趋势只在有限的变化点上改变增长率。一组 n 个 c 的变化点在不同时间定义为 $C = (c_1, c_2, ..., c_{n_c})$。在变化点之间，趋势增长率保持不变。第一段的增长率和偏移量分别给定为 δ_0 和 ρ_0。每个变化点的速率调整可以定义为一个向量 $\delta \in R^{n_c}$，其中 δ_j 是 j^{th} 变化点的速率变化。时间 t 的增长率是通过将初始增长率 δ_0 与截至时间步长 t 的所有变化点的速率调整之和相加来确定的。每个增长率变化 δ_j 是一个要在数据上拟合的参数。相应的偏移调整矢量可以定义为 $\rho \in R^{n_c}$。同样，时间 t 的偏移量由初始偏移量 ρ_0 和到时间 t 为止的每个变化点的偏移量调整之和给出。然而，变化点 c_j 的偏移量不是一个独立参数，而是定义为 $\rho_j = -c_j \delta_j$。这种特殊的偏移定义使得片断线性序列是连续的。此外，我们引入了一个二元向量 $\Gamma(t) \in R^{n_c}$，代表时间 t 是否已经过了每个变化点。因此，我们可以为时间 t 的趋势 $T(t)$ 定义一个矢量方程，在方程3中给出。

$$T(t) = (\delta_0 + \Gamma(t)^T \delta) \cdot t + (\rho_0 + \Gamma(t)^T \rho) \tag{3}$$

其中。

$$\delta = (\delta_1, \delta_2, ..., \delta_{n_c})$$

$$\rho = (\rho_1, \rho_2, ..., \rho_{n_c})$$

$$\Gamma(t) = (\Gamma_1(t), \Gamma_2(t), ..., \Gamma_{n_c}(t))$$

$$\Gamma_j(t) = \begin{cases} 1, & \text{如果 } t \geq c_j \\ 0, & \text{否则} \end{cases}$$

NeuralProphet提供了一个简单、半自动的机制来选择相关的变化点。给定所需变化点的数量 n_C，沿系列选择 n_C 等距的点作为初始变化点。在模型训练过程中，可以选择将它们的增长变化率参数正规化。这类似于全自动的变化点选择，因为只有最相关的变化点会被选中，如果有的话。用户也可以选择手动定义自定义数量的变化点的具体时间。为了避免在少量的最终点上过度拟合，最终趋势段（在最后一个变化点之后）被设定为较大的观测值集

(默认：训练数据的15%）。在对未观察到的未来进行预测时，最后的增长率被用来线性推断趋势。

2.1.2. 季节性

NeuralProphet中的季节性是通过使用傅里叶项（Harvey & Shephard, 1993）来处理的，就像最初在Prophet中做的那样（Taylor & Letham, 2017）。在这种技术中，为每个季节性定义了一些傅里叶项，如公式4，其中k指的是为具有周期性p的季节性定义的傅里叶项的数量。傅里叶项被定义为正弦、余弦对，允许对多种季节性以及具有非整数周期性的季节性进行建模，如具有每日数据（p=365.25）或每周数据（p=52.18）的年度季节性。在多季节性的情况下，可以为每个周期性定义不同的n值。

$$S_p(t) = \sum_{j=1}^{k} \left[a_j \cdot \cos\left(\frac{2\pi jt}{p}\right) + b_j \cdot \sin\left(\frac{2\pi jt}{p}\right) \right] \tag{4}$$

傅里叶项是模拟季节性的一个很好的工具，因为它们产生的平滑函数可以简单地解释并稳定地拟合数据。然而，傅里叶项只能模拟确定的季节性形状，这些形状被假定为在一段时间内是固定的。更多的傅里叶项可以使模型适应更复杂的季节性模式。太多的灵活性可能会导致过度拟合或观测之间的随机模式。因此，每个傅里叶项对应的频率与i，由正弦和余弦变换的加权组合来建模。每个季节性都与2k个系数有关。对于时间步长t，模型中考虑的所有p季节性的影响可以用公式5中的$S(t)$表示，其中P指的是所有周期性的集合。

$$S(t) = \sum_{p \in P} S_p^*(t) \tag{5}$$

加法和乘法的季节性模式都得到支持。每个季节性周期S^*_p可以单独配置为乘法，在这种情况下，分量要与趋势相乘。

$$S_p^*(t) = \begin{cases} T(t) \cdot S_p(t) & \text{如果} S_p \text{是乘法的} \\ S_p(t) & \text{否则} \end{cases}$$

该框架自动激活每日、每周和或每年的季节性，这取决于数据频率和长度。这三种类型的季节性周期性都被激活，如果

数据频率的分辨率高于各自的周期性，并且至少有两个完整的数据期。举例来说，如果数据的频率是每天，如果数据跨越两年或更长时间，该模型将启用年度季节性。如果有两个或更多星期的数据，每周的季节性也将被添加。每日的季节性将不会被激活，因为每日的频率没有更高的分辨率来允许日内的季节性。除非另有规定，每个季节性的傅里叶项的默认数量是：p=365.25年时k=6，p=7周时k=3，p=1日时k=6。

2.1.3. 自动回归

自动回归（AR）指的是将一个变量的未来值与它的过去值进行回归的过程，这是许多预测应用的一个关键部分，在Hyndman & Athanasopoulos（2014）中可以看到。包含的过去值的数量通常被称为AR（p）模型的阶数p。因此，每个过去的值都有一个系数θ_i。每个系数θ_i控制一个特定的过去值对预测的影响方向和大小。在一个经典的AR过程中，包括一个截距c和一个白噪声项E_t。

$$y_t = c + \sum_{i=1}^{i=p} \theta_i \cdot y_{t-i} + e_t$$

在许多应用中，我们对预测未来的多个步骤感兴趣。传统的AR模型只能对h=1的一步超前预测产生一个预测结果。如果需要一个长度为h>1的多步超前预测，就必须拟合h个模型，每个预测步骤一个。NP中的AR模块是基于Triebe, Laptev & Rajagopal（2019）中的AR-Net的修改版本。AR-Net能够用一个模型产生所有h个预测，该模型可以是线性或非线性的。在任何配置中，目标变量y_{t-1}，y_{t-2}，…，y_{t-p}，也被称为滞后期的p个最后观测值是模块的输入。输出是每个预测步骤$A^t(t), A^t(t+1), \ldots, A^t(t+h-1)$对应的AR效应的$h$值。因此，$A^t(t+2)$表示在$t+2$的预测$\hat{y}_{t+2}$的预测AR效应，这是未来的三个步骤，在预测原点$t$的预测，包括$t-1$的观察数据。

$$A^t(t), A^t(t+1), \ldots, A^t(t+h-1) = \text{AR-Net}(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) \quad (6)$$

值得注意的是，每次我们在一个特定的原点进行预测时，我们会得到h 个预测。因此，对于一个给定的时间点，我们有多达h 个不同的预测，每个都来自于过去的不同预测。根据预测时模型可用的数据，它们彼此不同。例如，当我们使用$AR(5)$模型预测未来的h=3步时，在给定的时间t=3，我们将有3个AR效应的预测值为

$\hat{y}_{t=3}$。这三个效应$A^{t-1}\ (t\ =\ 3)$, $A^{t-2}\ (t\ =\ 3)$, ..., $A^{t-3}\ (t\ =\ 3)$分别是对真实AR效应$A(t\ =\ 3)$的估计，具有不同的 "年龄"。最早的估计值$A^{t-1}\ (t\ =\ 3)$是3步的，而最近的估计值$A^{t-3}\ (t\ =\ 3)$是一步的。在自动回归中，任何估计值都至少是一步之遥，因为我们假设永远不知道系列在当前时间t的真实值，而只知道在最后一步t-1和更早的观察值。

AR 顺 序。这个模块最重要的参数是要回归的过去值的数量，也称为AR（p）模型的阶数p。这个参数应该根据过去观测中相关背景的大致长度来选择。在实践中，很难做到准确地确定，通常被设置为最内部周期的两倍或预测范围的两倍。另外，在与正则化结合使用时，可以选择一个保守的大阶，以获得一个稀疏的AR模型。

线 性 AR。默认的AR-Net配置不包含隐藏层，在功能上与经典AR模型相同。在实践中，它是一个单层NN，有p 个输入，h 个输出，没有偏置，也没有激活函数。单层权重各自将一个特定的滞后回归到一个特定的预测步骤。因此，每个权重可以与h 个经典AR(p)模型集合的相应系数相匹配，使模型的解释变得简单。

$$y = Wx$$

其中。

$$x = (y_{t-1}, y_{t-2}, ..., y_{t-p})$$

$$y = (A^t\ (t), A^t\ (t+1), ..., A^t\ (t+h-1))$$

我们可以把模型表示为预测的AR效应$y \in \mathbb{R}^h$ 的向量矩阵乘法，滞后观测值作为输入$x \in \mathbb{R}^p$

，权重矩阵$W \in \mathbb{R}^{h \times p}$，其中$W_{i,j}$表示定义滞后$y_j$对AR效应$A^t$（$t = i$）的线性影响的系数。

深 度 AR 。 基于AR-Net的AR模块可以对非线性动态进行建模，当隐藏层被计算出来时。在这种情况下，该模块训练一个完全连接的神经网络（NN），并将其与指定的
隐层和维度的数量。隐层的增加可能会导致更好的预测精度，然而它在可解释性方面是一个部分的权衡。我们不能直接量化某个过去的观察对某个预测的贡献，而只能观察某个过去的观察对所有预测的相对重要性。这可以通过比较第一层对每个输入位置的绝对权重的总和来近似。

时间序列的p 个最后观测值是第一层的输入。在每个隐藏层之后，logits通过一个激活函数，在我们的例子中，是一个整顿的线性单元（ReLU）。最后一层的输出是h logits，没有经过激活函数的转换，也没有偏置。对于l个隐藏层，隐藏层尺寸为d，我们有。

$$
\begin{aligned}
a_1 &= f_a\,(W_1\,x + b_1\,) \\
a_i &= f_a\,(W_i\,a_{i-1} + b_i\,) \text{ for } i \in [2, \ldots, l] \\
y &= W_{l+1}\,a_l
\end{aligned}
$$

其中。

$$
f_a(x) = ReLU\ (x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}
$$

因此，层偏置都有相同的尺寸$b \in \mathrm{R}^d$，而层权重是$W \in \mathrm{R}^{d \times d}$，除了第一个$W_1 \in \mathrm{R}^{d \times p}$和最后一个$W_{l+1} \in \mathrm{R}^{h \times d}$层权重。

稀 疏 的 AR 。 AR-Net表明，当正则化用于疏散模型权重时，可以通过将顺序设置为略大于预期值来近似地确定正确顺序。这使得模型的配置更加方便，同时保留了可解释性。

AR-Net的作者提出的正则化函数在方程8中给出。虽然我们提供了这个正则化函数，但在默

认情况下，我们的实现使用了一个不同的正则化函数，我们发现这个函数在更广泛的数据范围内效果更好。我们使用

方程14中引入的正则化函数，参数E=3，α=1，其中θ是对应于p个AR模型系数的权重向量。

$$\Lambda_{A}(\theta) = \Lambda(\theta, E = 3, \alpha = 1) = \frac{1}{p}\sum_{i=1}^{p} log(\frac{1}{3 - e^{|\theta_i|}}) + log(3) + 1 \tag{7}$$

其中。

$$\theta = \begin{cases} W_0 \in \mathbb{R}^{n_l}, & \text{无隐藏层 } (l = 0) \\ W_l \in \mathbb{R}^{n_l}, & \text{隐蔽层} (l \geq 1) \end{cases}$$

例如，在预测每小时时间序列的未来24步时，预测者可以选择将AR阶数设置为100，并进行一些正则化处理。拟合后，AR模块可能表现出稀疏的系数，最重要的权重仅位于几个位置上。在这个例子中，只有1、2、3、24、48和72号位置可能有非零权重。这些位置可以被解释为具有自相关的位置，此外，它们的权重可以被检查，以研究过去观察值的不同组合如何影响预测。

作为参考，在AR-Net中提出的原始正则化函数是由以下内容给出的。

$$\Lambda_{AR\text{-}Net}(\theta, c_1, c_2) = \frac{1}{p}\sum_{i=1}^{p} 2 - (1 + \exp(-c_1 \cdot |\theta_i|^{\frac{1}{c_2 - 1}}) - 1 \tag{8}$$

其中，作者建议设置$c_1 \approx 3$，$c_2 \approx 3$。

2.1.4. 滞后回归因子

滞后的回归者被用来将其他观测变量与我们的目标时间序列相关联。它们通常被称为协变量。与未来的回归者不同，滞后回归者的未来对我们来说是未知的。在预测的时间t，我们只能获得它们的观察值，过去的值，包括t-1。

$$L(t) = \sum_{x \in X} L_x(x_{t-1}, x_{t-2}, \ldots, x_{t-p}) \tag{9}$$

给定一组协变量$X \in R^{T \times n_l}$，我们为每个协变量创建一个单独的滞后回归者模块m个长度为T的协变量x。这样就可以单独确定每个协变量对预测的影响。每个滞后回归者模块在功能上与AR模块相同，唯一的区别是输入。这里，协变量x的最后p个观测值是模块的输入（而不是像AR中的序列y本身）。输出的形式是相同的，每个

模块产生h个添加成分$L^t(t), L^t(t+1), \ldots, L^t(t+h-1)$到整个预测中。

$\hat{y}, \hat{y}_{+1}, \ldots, \hat{y}_{+h-1}$。

$$L^t_x(t), L^t_x(t+1), \ldots, L^t_x(t+h-1) = \text{AR-Net}(x_{t-1}, x_{t-2}, \ldots, x_{t-p}) \quad (10)$$

关于每个协变量的AR-Net模块的顺序、深度、稀疏化和可解释性的所有考虑都与AR模块相同，如2.1.3节所讨论的，如果我们将其替代。

$$x = (x_{t-1}, x_{t-2}, \ldots, x_{t-p})$$
$$y = (L^t_x(t), L^t_x(t+1), \ldots, L^t_x(t+h-1))$$

2.1.5. 未来的回归者

为了对未来的回归者进行建模，这些回归者的过去和未来的数值都必须是已知的。给定未来回归者的集合为$F \in \mathbb{R}^{T \times n_f}$，其中$n_f$是回归者的数量，在时间步骤$t$的所有未来回归者的影响可以用$F(t)$表示，如公式11，其中$d_f$代表未来回归者$f \in V$的模型系数。默认情况下，未来回归者有一个加法效应，可以配置为乘法效应。

$$F(t) = \sum_{f \in F} F^*_f(t) \quad (11)$$

其中

$$F_f(t) = d_f f(t)$$

。

$F^{\dagger}(t) = T(t) - F_f(t)$。如果$f$是乘法的

$F^*_f(t) =$ $F_f(t)$。否则

2.1.6. 活动和节日

特殊事件或假期的影响可能是零星发生的。这类事件的建模与未来的回归者相似，每个事件e是一个二元变量$e \in [0, 1]$，表示该事件是否在特定的一天发生。对于一组事件$E \in \mathbb{R}^{T \times n_e}$，$n_e$事件的数量和序列的长度$T$，在时间步骤$t$的所有事件的影响可以用公式12中的$E(t)$表示，其中$z_e$表示事件$e \in E$对应的模型的系数。

$$E(t) = \sum_{e \in E} E_e^*(t) \quad (12)$$

其中

$$E_e^*(t) = \begin{cases} E_e(t) = z_e\, e(t) & \text{如果} e \text{是乘法的} \\ E_e(t)。 & \text{否则} \end{cases}$$

NeuralProphet允许对两种类型的事件进行建模：1）用户定义的2）国家特定的假期。给定一个国家的名称，它的国家假期会被自动检索并添加到事件集E中。与季节性效应类似，事件也可以被指定为加法或乘法。

此外，对于时间t_e的特定事件，可以配置一个$i+j$天的窗口$[t_e - i, t_e + j]$以被视为其自身的特殊事件。例如，通过设置一个$[-1, 0]$的窗口为圣诞节，将允许圣诞节前一天对预测有自己的影响。因此，为事件周围窗口内的每一天创建一个新的变量，并添加到事件集E中。

2.2. 预处理

2.2.1. 缺失的数据

在处理非滞后输入变量时，缺失数据的问题不大，因为相应的时间段可以简单地被放弃。这样做的话，每个缺失的条目会损失一个数据样本。在使用自动回归或滞后回归模块时，由于h个预测目标缺失和p个滞后值缺失，一个缺失的数据点会导致$h+p$个数据样本被丢弃。例如，对于一个使用AR(p=10)预测h=3步的模型来说，一个缺失点会导致13个样本的丢失。因此，我们实施了一个数据归因机制，以避免在处理不完整数据时的过度数据损失。该归因机制遵循以下启发式方法。

数据归纳。

如果没有指定或缺失，则假定事件没有发生。缺失的事件用零来填补，表示其不存在。所有其他实值回归变量，包括时间序列本身，如果启用了自回归，将通过三步程序进行估算。

首 *先* 。
缺失的数值是通过双向线性插值来接近的。因此，缺失值前后的最后一个和第一个已知值被用来作为内插的锚点。这是对每个方向上最多5个缺失值进行的。如果有超过10个缺失值，在这一步之后，它们将保持为*NAN*。要插值的缺失值的数量
是用户可设置的。

第 *二* 。
剩余的缺失值用中心滚动平均法进行估算。滚动平均数是在30个窗口中计算的，最多可以填补20个连续缺失值。用滚动平均数来填补缺失值的数量是用户可以设定的。

第三。如果有超过30个连续的缺失值，推算算法就会中止，而放弃所有的缺失数据点。

2.2.2. 数据规范化

应用于时间序列的归一化类型可由用户设定。可用的选项在表1中描述。如果没有指定，或设置为 "自动"，则使用默认选项 "软"，除非时间序列值是二进制的，在这种情况下，应用 "最小值"。

名称	归一化程序
'auto	"minmax'如果是二进制，则为'soft'。
'关闭'。	绕过数据规范化
'minmax'	将最小值缩放为0.0，将最大值缩放为1.0 'standardize'
	以零为中心，除以标准差
'软'。	将最小值缩放为0.0，将第95个四分位数缩放为1.0 'soft1'
	将最小值放大到0.1，将第90个四分位数放大到0.9

表1：可用的数据规范化选项。

2.2.3. 表格化

我们将时间序列数据表格化，以创建一个伪独立和相同分布的数据集，这也是基于SGD的训练所需要的。因此，为目标时间序列 "y

"的每个可用时间戳创建一个数据样本。一个样本包括一个归一化的时间戳，以及每个配置的模块所需的所有输入。

自动回归和滞后协变量模块从各自的时间序列中提取给定时间戳之前的p值，并将其存储在一个大小为p的向量中，作为模型的输入。如果提前多步预测（h>1），预测目标也被存储在每个时间戳*的*h*大小的*向量中。

这并不是一种准备训练数据集的内存效率高的方法。然而，我们选择这个程序是由于它的简单性和训练时的计算效率。

类似地，通过计算标准化时间分量的2k个不同的正弦和余弦变换形式，为每个配置的季节性周期准备傅里叶项。

2.3. 培训

Prophet和NeuralProphet的根本区别之一是拟合程序。Prophet利用L-BFGS（Nocedal & Wright，2006），在Stan（Carpenter等人，2017）中实现，将模型参数与数据进行拟合。NeuralProphet重新调整了Prophet，从底层开始，用PyTorch取代Stan，它既灵活又易于使用。

NeuralProphet依赖于现代版本的迷你批量随机梯度下降（SGD）。这种拟合程序与绝大多数深度学习方法兼容，可以可靠地扩展到大型数据集，并且可以容纳更复杂的模型组件。任何可由SGD训练的模型组件都可以作为一个模块被包含。这使得开发者很容易用新的功能扩展框架，并采用新的研究。

2.3.1. 损失功能

默认的损失函数是Huber损失，也被称为平滑L1损失，可以在公式13中看到。在一个给定的阈值β以下，它相当于平均平方误差（MSE）。对于大于β的值，它等同于平均绝对误差（MAE）。与纯粹的MSE损失相比，它对异常值不太敏感，可能有助于防止爆炸性梯度

Girshick（2015）。我们选择β=1作为阈值。然而，用户可以选择MSE或MAE损失，或任何其他可用的或自我实现的符合PyTorch损失函数格式的函数作为其损失函数。

$$L_{huber}(y, \hat{y}) = \begin{cases} \frac{1}{2\beta}(y-\hat{y})^2 。 & 对于|y-\hat{y}| < \beta \\ |y-\hat{y}| - \frac{\beta}{2} 。 & 否则 \end{cases} \quad (13)$$

2.3.2. 正规化

我们使用绝对权重值的缩放和移位的对数变换作为一般正则化函数。对于一个具有模型权重θ的模块，其参数化如下。

$$\Lambda(\theta, E, \alpha) = \frac{1}{n}\sum_{i=1}^{n} log(\frac{1}{E \cdot e} + \alpha \cdot |\theta_i|) + log(E) + 1 \quad (14)$$

其中，$E \in$（0，∞）设定偏移量的逆值，$\alpha \in$（0，∞）设定对数变换的缩放比例。E的值越大，权重接近零时，曲线越陡峭。参数E可以被描述为对权重稀疏化的*渴望的*控制。α的数值越大，权重越大的曲线就越平坦。参数α可以被解释为对大权重的*接受程度的*控制。作为默认值，我们建议E=1和α=1用于大多数应用。

$$\Lambda(\theta, E = 1, \alpha = 1) = \frac{1}{n}\sum_{i=1}^{n} log(\frac{1}{e} + |\theta_i|) + 1$$

正则化以每个模块配置的强度应用，并添加到损失函数中，以进行反向传播。正则化仅在训练的特定百分比后开始，默认情况下是在50%后，此后在训练结束时从零线性增加到其全部配置的强度。

2.3.3. 优化器

与损失函数类似，任何符合PyTorch优化器特征的优化器都可以被配置。作为一个可靠的默认值，除非另有说明，否则将使用AdamW优化器（Loshchilov & Hutter, 2019）。在我们的测试中，我们观察到AdamW是最可靠的拟合，有轻微的过拟合倾向。我们用配置的学习率初始化AdamW，并设置β=（0.9，0.999），E=1e - 08，以及1e - 04的权重衰减。

作为后备选项，我们还提供了一个经典的SGD优化器，动量设置为0.9，权重衰减设置为1e - 4。在我们的测试中，我们发现SGD能带来更好的验证性能，但代价是偶尔会出现分歧。如果用户愿意对训练相关的超参数进行微调，SGD是AdamW的一个有吸引力的选择。

2.3.4. 学习率

由于我们不想要求用户成为机器学习方面的专家，我们把学习率这个任何NN都必须

的超参数变成了可选项。我们通过依靠一个简单的

但合理有效的估计学习率的方法，在Smith（2017）中作为学习率范围测试引入。

学习率范围测试被执行了$100 + log_{10}(10 + T) * 50)$次迭代，开始于$\eta = 1e-7$，在$\eta = 1e+2$时结束，配置的批次大小。每次迭代后，学习速率呈指数增长，直到最后一次迭代中达到最终的学习速率。不包括前10次和最后5次迭代，最陡峭的学习率被定义为最终的学习率。最陡峭的学习率是通过选择使损失的负梯度最大化的位置的学习率而找到的。

为了提高可靠性，我们进行了三次测试，并将三次运行的对数10平均值η_1，η_2，η_3作为选定的学习率η。

$$\eta = \frac{1}{3}(log_{10}(\eta_1) + log_{10}(\eta_2) + log_{10}(\eta_3))$$
$$\eta = 10^{\eta^*}$$

2.3.5. 批量大小

批量大小B是一个可选的参数。如果没有用户指定，下面的启发式方法将根据数据集的长度T来确定批处理的大小。

$$B^* = 2^{2+\lfloor \log_{10}(T) \rfloor}$$
$$B = \min(T, \max(16, \min(256, B^*)))$$

2.3.6. 训练纪元

训练纪元数N_{epoch}是一个可选参数。 如果用户没有指定，下面的启发式方法将决定 epochs 的数量。

$$N_{纪元} = \frac{1000 - 25}{\log_{10}(T)}$$
$$N_{epoch} = \min(500, \max(50, \lfloor N \rfloor_{纪元}^*))$$

2.3.7. 节目表

由于所有与训练有关的超参数都是自动近似的，没有一个是最优的，因此可能会潜在地

导致训练问题。尽管如此，用户还是希望该模型

以合理地拟合数据而没有问题。为了满足这些期望，尽管有次优的超参数，我们依靠称为"1cycle "政策的训练时间表，根据Smith & Topin（2018），它允许NN训练的"超级收敛"。因此，初始

学习率 η_{100} 逐渐增加，直到训练的30%时达到峰值学习率η。

此后，学习率沿着余弦曲线退火，在训练结束时下降到 η_{5000}。

2.4. 后期处理

在训练或预测结束时，模型所操作的归一化数值被转换回初始分布。这与许多启发式或程序式设置的可选超参数一起，抽象了许多机器学习的具体决策和时间序列的具体决策，使NeuralProphet对预测从业者来说很容易接近。

2.4.1. 度量衡

有几个指标是默认记录的。配置的损失函数、RMSE和MAE。此外，任何用户定义的指标也将被记录在训练和验证运行中。

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(y_i - \hat{y}_i)^2} \tag{15}$$

$$MAE = \frac{1}{T}\sum_{i=1}^{T}|y_i - \hat{y}_i| \tag{16}$$

其中

T = 数据的大小

2.4.2. 预测介绍

当用于预测时，该模型会返回一个数据框架，其中有一列是关于每个预测成分的。这些是每个成分的加法（或乘法）贡献。它们被单独显示在每个预测步骤的前面。每个成分都是指其行的日期戳定位的目标，并按预测的年龄排序。例如，'yhat3'是指当前位置的预测*Y值*，基于三步前的数据。同样，'ar2'指的是根据两步前的数据，对当前位置的AR效

应进行预测。

有不同的绘图工具可用于可视化预测本身、预测分解、模型参数，以及感兴趣的特定预测范围。

3. 实验设置

这项工作中的实验是为了对比NeuralProphet和Prophet的能力。首先，我们在一组合成数据集上比较了先知和神经-
先知的可解释预测部分的准确性。其次，我们在一些真实世界的数据集上比较这两个框架的性能。在下文中，我们将详细描述这些实验。

3.1. 合成数据上的分解演示

这些实验对比了Prophet和NeuralProphet将一个时间序列分解为各个组成部分的能力。由于我们对不同建模成分的分解准确性感兴趣，我们必须知道它们的基础真相。由于缺乏合适的具有已知基础成分的真实世界数据集，我们创建了合成数据集，作为添加了噪声的生成的时间序列的总和。这使我们能够将两个模型的估计分解成分与原始成分进行比较。

3.1.1. 合成数据集的构成

我们生成的单个成分时间序列与Prophet和NeuralProphet声称要建立模型的成分类型相同。因此，在理想情况下，这两个模型应该能够将时间序列完美地分解为其组成部分，但由于合成数据集中包含的加性噪声而产生一些误差。

在表2中，我们显示了每个实验场景的合成数据集中包括哪些成分。两种模型之间的比较集中在单个分解成分的准确性上。我们还测量了总体精度和计算时间。因此，重点是单个组件的准确性，因为这可以量化模型的分解准确性。

由于许多现实世界的数据集表现出自动回归的特性或依赖于滞后的回归者，我们还包括三个带有滞后成分的实验。然而，只有NeuralProphet声称能够对自动回归和滞后回归者成分建模。在包括这种滞后成分的实验中，Prophet对这些成分的预测将被标记为零。

实验	趋势	季节性	活动	未来注册。	AR	滞后调节。
S-TS	X	X				
S-EF			X	X		
S-TSEF	X	X	X	X		
S-mTSEF	X	~~X†~~	~~X†~~	~~X†~~		
S-AL					X	X
S-TSAL	X	X			X	X
S-TSEFAL	X	X	X	X	X	X

表2：实验方案中的合成成分。†表示各自的成分与趋势相乘。

由于可以自由选择其加法分量中的偏移量，模型可能会找到与基础分量的动态相匹配的可预测分量，但被偏移了。因此，在计算任何分量上的度量之前，我们对所有分量进行零中心化。

3.1.2. 生成的成分时间序列

每个实验的数据集由5个独立生成的时间序列组成，每个时间序列的长度为6000个样本的日分辨率。每个时间序列都是由多个生成的时间序列聚合而成的，每个时间序列代表一个特定的成分类型。一个组件的时间序列在与其他组件聚合之前被缩放到范围[0, 1]。聚合的时间序列最后被重新缩放到[0, 1]的范围。为了表示合成实验的构件，我们使用以下符号，每个构件用一个字母表示。

T *：趋势*。一个有一个随机变化点的片状线性趋势。趋势在变化点之前线性增加，在变化点之后线性减少。

S *：季节性*。生成的季节性分量时间序列由与随机权重相结合的傅里叶项组成。分量是根据方程4生成的，a_j 和b_j 从随机均匀分布中抽取，这样$a_j \in [0, 1]$和$b_j \in [0, 1]$，$k = 5$。在实验中，我们创建了两个独立的季节性成分：月$S_{30}(t)$和年$S_{365}(t)$。两者都被添加到总序列中，并对两者的分解精度进行单独评估。

$\boldsymbol{E}$：*事件*。事件部分是一个二进制序列，有25个随机出现的事件
地点。与方程12类似，我们有$\boldsymbol{E} = (e^{1}) \in \mathrm{R}^{T\times 1}$，其中$e^{1} = (e^{1}_{1}, \ldots, e^{1}_{T}) \in \mathrm{R}^{T}$，其中
$e^{1}_{i} = 1$，如果在时间戳t_i发生e^{1}类型的事件，否则为0。初始效应强度
$z_e = 1$在缩放主系列后将有效地成为$z_e \geq 1$。*N组件*

$\boldsymbol{F}$：*未来调节器*。未来的调节器成分被生成为一个AR(3)过程，其共同效率为$\varphi_1 = 0.2, \varphi_2 = 0.3, \varphi_3 = -0.5$和加性白噪声。在将序列缩放到[0, 1]的范围内后，回归器被添加到总序列中，其权重为$d_f = 1$，在缩放主序列后将是$d_f \geq 1$。*N组件*

$\boldsymbol{A}$：*自动回归*。自回归成分来自AR(2)过程，系数$\varphi_1 = 0.3$，$\varphi_2 = 0.3$，以及加性白噪声。这个序列是独立于其他成分产生的。这与NeuralProphet对自动回归行为的建模方式不尽相同。NeuralProphet没有分解自动回归模块的输入，而是使用原始时间序列值作为输入。

$\boldsymbol{L}$：*滞后的回归者*。首先，用AR(2)过程生成一个独立的时间序列x，类似于未来的回归者。接下来，我们创建滞后的回归者效应序列$L(t) \in \mathrm{R}^{T}$
，其中每个效应取决于序列x的最后三个观测值的加权组合。

$$L(t) = c_1 \cdot x_{t-1} + c_2 \cdot x_{t-2} + c_3 \cdot x_{t-3}$$

其中，c_1，c_2，…，c_{lag}是来自（0，1）上的随机均匀分布的权重。最后，我们利用序列$L(t)$作为滞后的回归者成分。

$\boldsymbol{m}$：*乘法模式*。在乘法模式下，我们将每个成分逐一乘以趋势。例如，在S-mTSEF中。

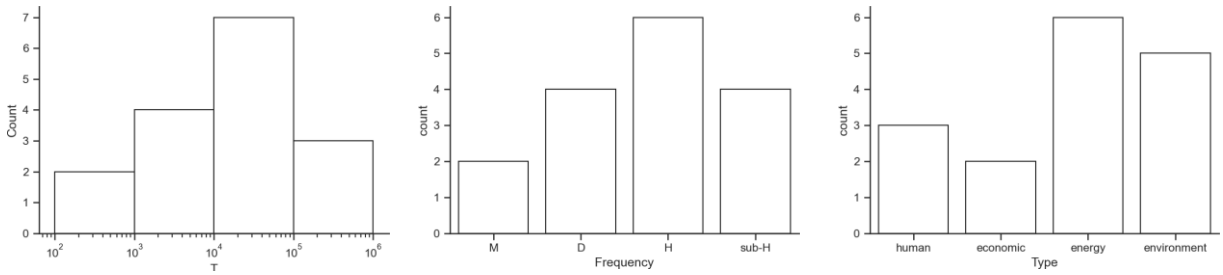
$$y_t = T(t) + E^{\dagger}(t) + S^{\dagger}(t) + F^{\dagger}(t))$$
$$= T(t) \cdot (1 + E(t) + S(t) + F(t))$$

3.2. 真实世界数据集的主要基准

我们在涵盖许多单变量预测任务的不同数据集上评估了先知和神经先知的预测准确性。这些数据集的数据长度从一个

100到50万个样本，而数据记录的时间间隔从分钟到月度的频率。这种广泛的单变量时间序列与大多数侧重于基于NN的方法的基准测试工作形成了鲜明的对比。这些结果代表了能源需求、旅游业、环境因素和零售业等应用的实际开箱即用的性能。

在本节中，我们将更详细地介绍所选的数据集和评估程序。



(a) 长度的直方图。 (b) 采样频率的计数。 (c) 数据类型的计数。

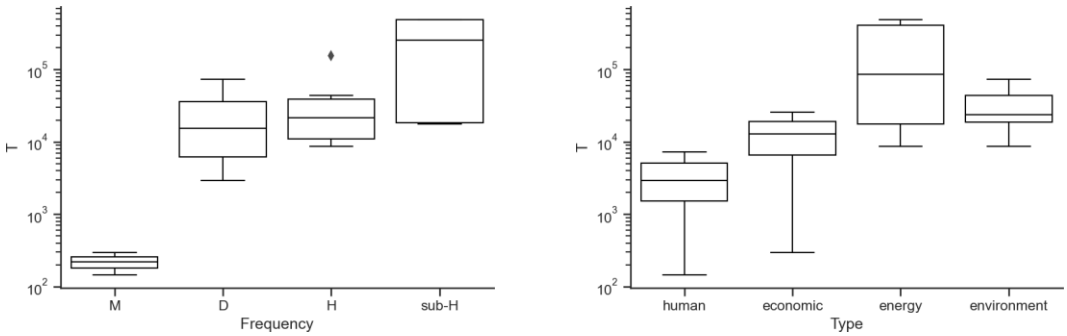
图1：数据集的不同特征的直方图。

3.2.1. 部分数据集

选择这些数据集是为了代表广泛的单变量数据集。图1a显示，数据集的长度从一百个到一百万个样本不等，大多数数据集都有一万个左右的样本。数据观察频率从月度到分钟级的采样频率都有分布。图1b将分钟级到半小时级的频率分在一起，并显示每类频率的数量，其中小时级的频率是最常见的。

我们收集的测试数据集涵盖了各种各样的主题，包括环境、能源相关、社会和经济数据，其中环境和能源相关的数据集最为常见（图1c）。

纳入我们收集的测试数据集的量化标准是。采样频率为分钟到日的单变量时间序列，长度为1000个样本或更多，以及允许的许可。我们还包括先知作为示范时间序列使用的四个时间序列。其中两个数据集不符合纳入标准，因为它们的采样频率是每月一次，长度远小于1000个样本。尽管如此，它们还是被包括在内，以确保这些数据集不偏向于拟议的模型NeuralProphet，而是偏向于比较模型Prophet。关于每个数据集的进一步细节可以在表8（附录A）中找到。



(a)数据频率的长度扩张图。 (b)数据类型的长度的波谱图。

图2：不同数据频率和数据类型下的数据长度分布。

资料来源。 数据集来自各种来源，包括UCI机器学习库（Dua & Graff，2017）、莫纳什时间序列库（Godahewa, Bergmeir, Webb, Hyndman & Montero-Manso, 2021）和先知Github库（Taylor & Letham, 2017）。此外，我们还从公共能源和政府网页上收集和处理数据，包括DOE（2015）、ERCOT（2021）、RTE（2018）和OpenEI（2015）。

预处理。 大多数选定的数据集不包含缺失值，但对于那些缺失值，除非另有说明，否则我们使用线性内插法进行估算。然而，对于那些大部分为零或接近零值的数据集，如涵盖太阳能、停车、风能和空气质量等主题的数据，我们用零来填补缺失值。缺失的时间戳也被认为是缺失的数据。此外，我们对每个数据集进行手工预处理，并修复微小的错位，如每小时采样频率的数据上的时间戳是01:59而不是02:00。

3.2.2. 基准模型

在这项工作中，我们比较了时间序列预测包的性能，这些预测包支持完全的模型自动化，提供可解释性，并可扩展到大型数据集，而不需要机器学习或时间序列的领域知识。Prophet是第一个这样的软件包，我们用它唯一功能齐全的后继者NeuralProphet作为基准。

超参数。 鉴于我们的目标是展示一个非专业的预测从业者可以合理地期望的真实世界的性能，我们没

有手动调整任何超参数。我们

如果没有自动设置，或者需要使模型具有可比性，则仅在合成基准测试部分手动设置超参数。

在讨论结果时，我们明确提到了任何手动设置的参数。出现手动配置模型的情况，但不是超参数调整，包括。当研究在模型中加入自动回归的效果时，我们设置了不同的滞后期数p，这些都在结果中显示。在反向消融研究中，我们进一步研究了在模型中加入隐藏层的影响，据此我们还将所有模型配置和所有数据集的学习率设置为同一数值0.001。

这些对滞后数、预测数和神经网络配置的消减研究也将有助于量化NeuralProphet对参数选择的敏感性。

3.3. 预测评估

在下文中，我们将描述预测任务以及如何评估预测性能。

3.3.1. 预测范围

在这项研究中，我们对涵盖典型预测任务的单变量时间序列预测感兴趣。给定预测未来的所需步数，我们从现在的时间戳开始，预测系列的未来几个连续值。从最后一个观察值开始预测的下一个连续值的数量也被称为预测范围。如果配置了自动回归，模型将对系列本身的最后几个观测值进行回归。我们把每个给定的预测范围作为一个单独的预测任务。

一个有10个目标的预测任务涉及到预测未来10个目标的开始。在当前的时间戳t，即获得一系列$\{\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+9}\}$，包含预测范围内的预测值。在我们对现实世界数据集的经验评估中，我们评估了[∞，1，3，15，60]的预测期限。因此，∞的水平线指的是一次性预测整个测试集，就像只基于时间特征的模型一样，例如先知和默认的神经先知。

3.3.2. 扩大原点回测

时间序列预测的评估程序在本质上是多样的。命名惯例只为经典时间序列模型的评估而建立（Tashman (2000)）。目前正在建立评估基于机器学习的时间序列预测模型的惯例。我们利用一种基于交叉验证和回溯测试的方法，这种方法起源于

机器学习和交易算法评估。这种方法被称为扩大起源回测，最近被试图在时间序列数据上评估基于机器学习的模型的研究人员和公司采用（Uber（2018，2020））。

该程序由一个外循环和一个内循环组成。扩大原点回测这个术语主要是指外循环，但经常被用来描述整个程序。内循环通常是基于经典评估程序的一个变体。在我们的例子中，内循环是一个滚动的原点预测评估，具有非恒定的保持规模（Svetunkov, 2021），具有更新功能，但没有重新匹配模型。在经典的时间序列文献中，重新拟合被称为重新参数化，而更新模型是指更新模型的输入，以产生下一个预测，而不重新拟合模型参数。这意味着一个模型在每个评估程序中只被重新拟合了k次。

我们的预测评估程序的一个更详细的名称可以是。K-fold expanding origin backtest with an inner rolling origin multi-step forecast evaluation with updating but without refitting of the model.

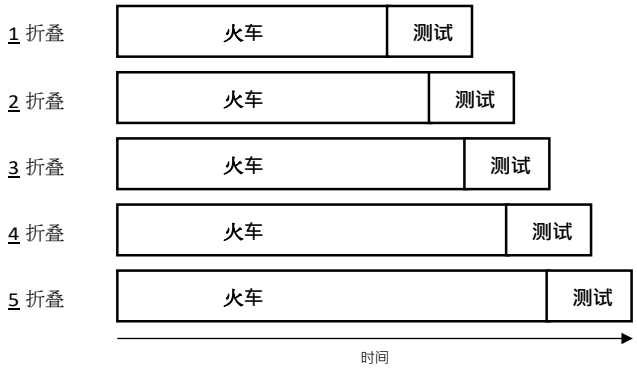


图3：扩大原点的回测，有5个折页。测试集各包含10%的时间序列。

*外循环。K-折扩大原点回测，保持大小不变。*类似于k-*fold*交叉验证，整个数据集被复制成k个折叠，每个折叠包含训练和测试数据，覆盖整个数据集的不同部分。要评估的模型在每个折叠的训练数据上单独拟合，并在同一折叠的测试数据上进行评估。这标志着该模型

与k-*fold*交叉验证不同，折叠不是随机的。每个折子都有一个训练集，其数值严格地发生在同一折子的测试集之前，图3中可以看到一个可视化的结果。

我们用10%的测试集进行了5次扩大原点的回测，并在每个折页上对原点进行了5%的正向滚动。这意味着每个测试折页将与前一个折页共享50%的数据，每个训练折页将增长5%，而测试折页的规模保持不变。因此，第一个折叠将使用前70%的数据进行训练，接下来的10%用于测试。第五折将利用90%的数据进行训练，最后10%的数据进行测试。因此，一个模型要在10%的数据上测试五次，覆盖所有测试中最后30%的数据。

内环。滚动起源与更新，但没有重新装配。 内循环对每个训练-测试折线进行重复。首先，对训练集进行模型拟合。测试集的评估与具有非恒定保持量的滚动原点预测评估相同（Svetunkov，2021），具有更新功能，但没有重新拟合模型。

因此，预测原点每次向前滚动1步，不需要用新数据重新训练模型。这在图3所示的单一测试折线中重复进行，直到测试折线结束。依靠滞后观测值作为输入的自动回归模型用新的数据点更新以产生下一个预测。如果一个模型在每个预测原点产生多个预测，所有预测步骤的平均准确度被记录下来。对于具有无限预测步骤的模型，不依赖于被更新的新观测值，我们对整个测试集进行一次预测。

在Hyndman & Athanasopoulos (2014)中，这个程序的单一预测版本也被称为时间序列交叉验证，我们认为这个命名很不幸，因为它不能与外循环（Expanding Origin Backtest）混淆，后者与*k-fold*交叉验证的关系更为密切。

3.3.3. 度量衡

我们计算每一个i步预测与相应的实际值的度量，然后在所有的步骤中平均这个误差。对于评估，我们使用两个指标。平均绝对比例误差（MASE）和平均平方误差（RMSSE），如Hyndman & Koehler（2006）所定义的。MSSE是指被评估方法的RMSE除以Na¨ve预测的RMSE。这是平均绝对比例误差（MASE）的平方类似物。因此，Na¨ve指的是预测下一个观察值与之前的观察值相同。

MASE和RMSSE指标允许直接评估一种方法与Na¨ve预测相比的性能，因为一个较小的值意味着比Na¨ve的改进。此外，由于这些指标是按比例的数量，不同的模型可以在不同的数据集中进行比较。

$$MASE = \frac{\frac{1}{J}\sum_{j=T+1}^{LT+J} |y_i - \hat{y}_i|}{\frac{1}{T-1}\sum_{i=2}^{LT} |y_i - y_{i-1}|} \tag{17}$$

$$RMSSE = \frac{\sqrt{\frac{1}{J}\sum_{j=T+1}^{LT+J} (y_i - \hat{y}_i)^2}}{\sqrt{\frac{1}{T-1}\sum_{i=2}^{LT} (y_i - y_{i-1})^2}} \tag{18}$$

其中

T = 训练数据的大小

J = 测试数据的大小

3.4. 计算资源

除非另有提及，所有的模型都是在一台带有8核CPU的普通笔记本电脑上训练和评估的（2020年的Macbook Pro，带有M1芯片）。

4. 结果

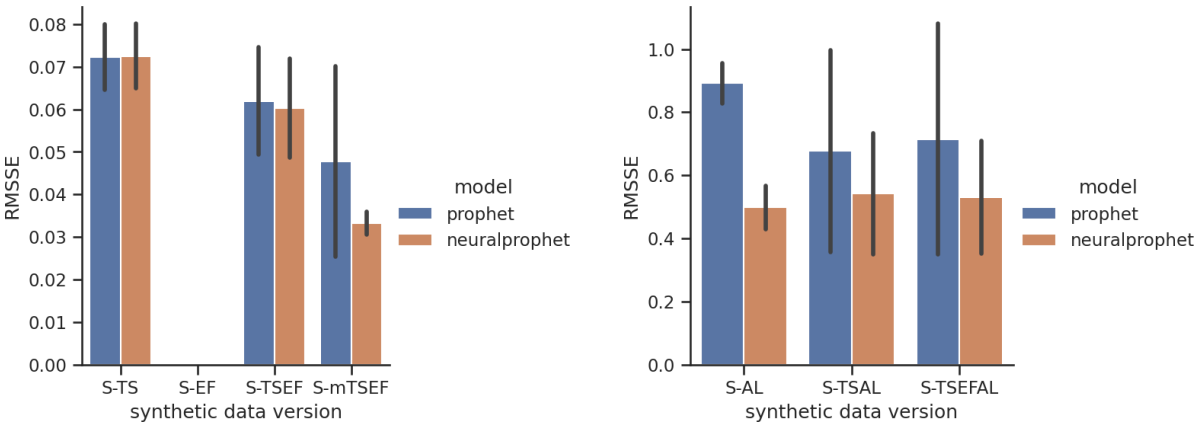
4.1. 可解释成分分解的演示

结果分别显示了实验S-TS、S-EF、S-TSEF和S-mTSEF（不含滞后成分）以及实验S-AL、S-TSAL和S-TSEFAL（包括滞后成分）的情况。在前者中，我们希望两个模型的表现相同，而在后者中，我们希望NeuralProphet的表现更好。

在没有滞后成分的数据上，NeuralProphet的表现与Prophet相同或略胜一筹，从图4和图5a可以看出。对"y"分量（分量之和）的总体拟合度相当或略好。最大的差异体现在具有乘法成分的*S-mTSEF*上，NeuralProphet的误差和误差标准差明显较低。

在涉及滞后成分的实验中，NeuralProphet的表现明显优于Prophet，如图4和图5a所示。大

部分的差异来自于对滞后回归因子更好的建模。



(a)没有滞后成分的实验。

(b) 包括滞后成分的实验。

图4：模型对合成时间序列y本身的拟合的RMSSE误差。图中条形显示平均值，线显示5次运行的标准差。

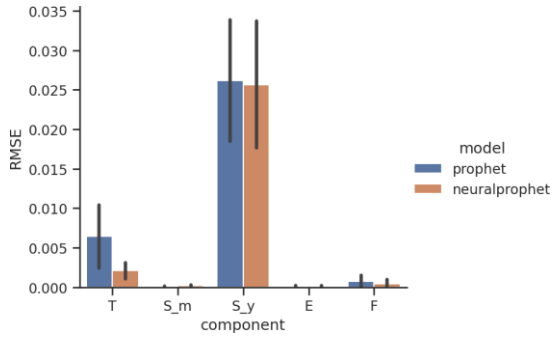
对序列本身的拟合并不像分量上的分解精度那样重要，因为整体的拟合度并不能防止过度拟合。分量上的准确性，是一个充分的衡量标准，因为一个能够捕捉到真实的基本动态的模型会有很好的概括性。

4.1.1. 组成部分分解的准确性

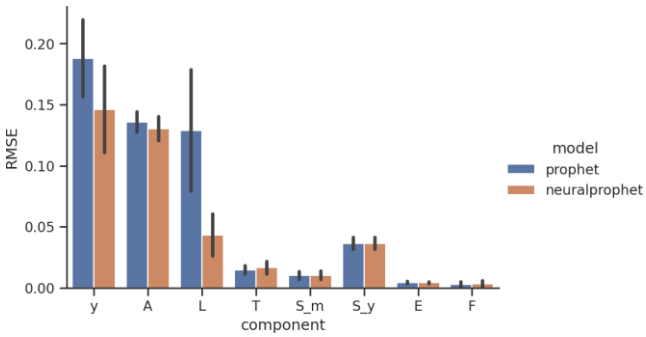
无滞后成分。从图5a中可以看出，当比较分量上的准确性时，NeuralProphet的表现与Prophet相同或更好。两种模型都能近乎完美地拟合月度季节性、事件和未来回归因子。两种模型都很难找到对年度季节性的良好拟合。这可能是由于数据中可用于拟合模型的完整年度的数量有限。在趋势部分的拟合度方面，观察到一个明显的差异，Neuralprophet表现出先知的一小部分误差。

包括滞后成分。在对滞后成分进行建模时，NeuralProphet在分解滞后回归因子方面有了明显的改进。然而，AR成分并没有比Prophet的零预测明显更准确。这是由于，NeuralProphet没有分解AR模块的输入，而是使用原始的时间序列值，而在这个实验中，我们基于独立添加的AR过程组件来评估性能。在实际应用中，这样的区别在很大程度上并不重要。

放大图7（附录B）中各个实验装置的性能，我们发现



(a)没有滞后成分的实验。



(b)包括滞后成分的实验。 *

图5：预测成分与潜在的真实生成成分值的成分间的RMSE。图中条形显示的是平均值，直线显示的是所有合成数据集和5次运行的标准偏差。*注：Prophet不支持自动回归和滞后调节器成分。Prophet对上述成分的估计隐含地假定为零。因此，Prophet显示的误差等同于成分的标准差。

与我们已经讨论过的图4和图5a相比，没有发现明显的新观察结果。

4.1.2. 计算时间

表3显示，与Prophet相比，NeuralProphet的训练平均慢了4.0倍。然而，两者都表现出较大的标准差，Prophet有一个向上的长尾巴，NeuralProphet则向下。预测时间表现出相反的关系，NeuralProphet的计算速度比Prophet快13.5倍。如图6所示，NeuralProphet的预测时间分布紧密，没有向上的异常值，而Prophet有一个大的标准偏差，尾巴向上很长。

模型	培训时间 (秒)	预测时间 (秒)
神经先知	20.50 (±4.70)	**0.16** (±0.05)
先知	**5.07** (±2.30)	2.16 (±1.08)

表3：训练模型的计算时间（秒）和整个数据集的预测时间。显示了在没有滞后成分的实验中所有折叠的平均值和标准偏差。

这些观察结果意味着NeuralProphet可能需要一些额外的资源来适应，但在部署时需要的资源将大大减少。此外，由于NeuralProphet在计算预测方面的速度快了一个数量级，这使得它有可能被部署在对时间敏感的应用中，在这种应用中，下一个预测需要在几分之一秒内被可靠地计算出来。

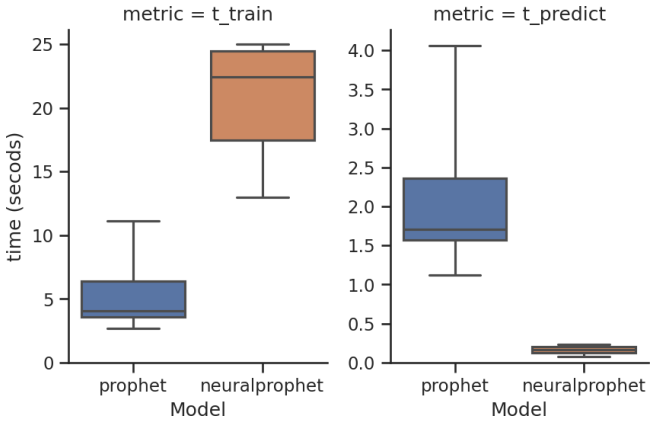


图6：在没有滞后成分的实验中，所有褶皱的训练和预测时间。

由于完全在PyTorch中实现，NeuralProphet模型在未来可能会被并行化，使其能够被部署在GPU上。我们希望这能缩小训练时间上的差距，而预测时间上的差距则会扩大。

4.2. 主要基准测试结果

总体结果显示在表4中。我们想强调三个观点。首先，Prophet和NeuralProphet在其默认模式下的表现几乎相同。然而，两者的表现都明显不如Naive领先一步。第二，当配置任何数量的滞后期时，NeuralProphet的预测都有很大的改善，一直比Naive预测提前一步的表现好。更多的滞后期通常会导致更好的性能。第三，当提前多步预测时，即使我们将水平线提高到60步，任何滞后数的NeuralProphet仍然比默认配置的Prophet或NeuralProphet表现得更好。

我们还观察到，所选择的具体滞后期数对预测精度的影响很小，这表明该模型对最佳参数选择并不敏感。

所有为任何模型设置的模型参数都明确显示在表4中：预测步骤数和滞后数。没有手动设置或调整超参数。

模型	不同预测期的MASE			
	1步	3步	15步	60步
先知[*1]		**8.54** (±2.17)		
NeuralProphet[*1]		**8.49** (±2.03)		
神经先知 (1滞后)	0.83 (±0.09)	不适用	不适用	不适用
神经先知 (3个滞后)	0.72 (±0.07)	不适用	不适用	不适用
神经先知（12个滞后）。	**0.69** (±0.07)	1.16 (±0.16)	不适用	不适用
神经先知(30滞后)	**0.62** (±0.07)	**0.99** (±0.12)	**2.07** (±0.30)	不适用
神经先知 (120滞后)[*2]	**0.62** (±0.09)	**0.94** (±0.12)	**1.97** (±0.29)	**3.77** (±0.81)

表4：所有数据集的每个预测范围的预测误差（MASE）。标准差显示在括号内。相应的RMSSE值在表9中给出（附录C）。注*1：无滞后期的模型可以产生任意数量的预测。注*2 ：由于滞后期为120的模型长度较短，所以没有对月度数据集进行评估。

选择MASE或RMSSE作为误差的衡量标准，对于单步预测任务来说是完全足够的，但对于多步预测来说却不是，因为它只衡量天真模型的单步预测误差。当提前多步预测时，MASE值为1不再等于天真模型的性能，因为天真模型在提前多步时可能会有更高的误差。尽管如此，MASE误差仍然是一个有用的数量，可以比较不同模型在不同数据集上的预测性能，因为它将误差放大到一个有点标准化的空间。

4.2.1. 消融研究。神经网络的深度

我们进一步研究了不同的神经网络配置选项对不同预测范围内的预测准确性的影响。总的来说，大多数配置的表现都在其线性对应物的范围之内。除了最长的60步预测范围外，所有的NN配置都明显优于其线性对应物。NN参数的具体选择对预测精度没有明显的影响。因

此，该模型对非最佳的NN参数选择不敏感，对于短期预测，也对缺乏任何隐藏层不敏感。

对于这项消融研究和任何其他提到的涉及NN的配置，学习

模型	不同预测期的MASE			
	1个步骤	3个步骤	15个步骤	60步
(30个滞后)	0.62 (±*0*.07)	0.99 (±*0*.12)	2.07 (±*0*.30)	不适用
(30个滞后，1x32 NN)	0.60 (±*0*.08)	0.93 (±*0*.12)	**1.88** (±*0*.23)	不适用
(30滞后，2x24 NN)	**0.59** (±*0*.07)	**0.91** (±*0*.13)	1.89 (±*0*.29)	不适用
(30个滞后，4x16 NN)	0.61 (±*0*.10)	0.93 (±*0*.12)	1.91 (±*0*.26)	不适用
(120个滞后)	**0.62** (±*0*.09)	0.94 (±*0*.12)	1.97 (±*0*.29)	3.77 (±*0*.81)
(120滞后，1x32 NN)	0.63 (±*0*.11)	0.92 (±*0*.13)	**1.81** (±*0*.24)	3.06 (±*0*.52)
(120滞后，2x24 NN)	0.69 (±*0*.14)	**0.90** (±*0*.12)	**1.81** (±*0*.22)	3.07 (±*0*.57)
(120滞后，4x16 NN)	0.66 (±*0*.11)	0.96 (±*0*.15)	1.86 (±*0*.31)	**3.03** (±*0*.47)

表5：NeuralProphet（含NN）对所有数据集的每个预测期的预测误差（MASE）。相应的RMSSE值见表10（附录C）。月度数据集被排除在外，因为它们没有足够的样本量来适应NN的参数。标准差显示在括号内。模型定义显示了输入滞后数和层数，包括它们的隐藏尺寸。例如，"4x16 NN "表示模型被设置为使用4个维度为16的隐藏层。注：对于所有带有NN的模型，学习率是手动设置的，但不是调整的，为0.001的猜测值。

对于所有的模型变化和数据集，比率被手动设置为0.001的值。这个值是一个猜测，没有经过调整。它的设置是为了避免来自学习率范围测试的潜在变化源。所有其他为任何模型设置的模型参数都明确显示在表5中：预测步骤数、滞后数、隐藏层数和隐藏层尺寸。

4.2.2. 基于数据主题类型的结果

一般来说，滞后期较长的模型往往表现更好。这一点在能源主题类型的数据集中尤其明显，120滞后期的NeuralProphet模型在1步超前预测和60步超前预测中分别减少了96%和66%的预测误差。此外，我们观察到，配置有4层16维NN的模型在1步预测和部分3步预测中的表现优于线性模型，而在所有数据类型的15步和60步预测中表现最佳。

总的来说，配置任何数量的滞后期都能显著提高NeuralProphet在任何预测范围内的预测性能。 除了在人类数据集上的60步水平线之外

	人	经济	环境	能源
		∞阶梯式MASE		
先知神经先知	1.11 (±0.26)	4.25 (±3.09)	9.27 (±3.89)	11.13 (±1.22)
	1.13 (±0.21)	2.39 (±0.85)	9.48 (±3.75)	11.13 (±1.40)
		1步式MASE		
神经先知 (1滞后)	0.76 (±0.08)	1.01 (±0.21)	0.81 (±0.10)	0.83 (±0.05)
神经先知 (3个滞后)	0.75 (±0.08)	0.98 (±0.14)	0.76 (±0.09)	0.63 (±0.05)
神经先知（12个滞后）。	0.66 (±0.07)	0.97 (±0.17)	0.75 (±0.08)	0.60 (±0.04)
神经先知(30滞后)	**0.62** (±0.07)	0.82 (±0.18)	**0.74** (±0.10)	0.48 (±0.03)
神经先知 (120滞后)*	0.67 (±0.15)	**0.75** (±0.12)	0.75 (±0.11)	**0.46** (±0.04)
神经先知（120滞后，4x16 NN）。*	0.75 (±0.13)	0.77 (±0.06)	0.83 (±0.16)	0.48 (±0.07)
		三步法MASE		
神经先知（12个滞后）。	0.77 (±0.09)	1.81 (±0.60)	1.25 (±0.17)	1.10 (±0.09)
神经先知(30滞后)	**0.73** (±0.09)	1.08 (±0.20)	1.21 (±0.18)	0.88 (±0.06)
神经先知 (120滞后)*	**0.73** (±0.11)	0.98 (±0.13)	**1.19** (±0.18)	0.79 (±0.06)
神经先知（120滞后，4x16 NN）。*	0.87 (±0.13)	**0.83** (±0.09)	1.27 (±0.25)	**0.76** (±0.08)
		15个步骤的MASE		
神经先知(30滞后)	0.89 (±0.16)	1.58 (±0.33)	2.52 (±0.48)	2.17 (±0.18)
神经先知 (120滞后)*	0.93 (±0.21)	1.30 (±0.30)	2.45 (±0.46)	2.04 (±0.18)
神经先知（120滞后，4x16 NN）。*	**0.88** (±0.16)	**1.18** (±0.10)	**2.40** (±0.57)	**1.85** (±0.19)
		60步 MASE		
神经先知 (120滞后)*	1.38 (±0.38)	1.70 (±0.43)	5.14 (±1.50)	3.78 (±0.43)
神经先知（120滞后，4x16 NN）。*	**1.05** (±0.34)	**1.52** (±0.12)	**3.75** (±0.81)	**3.34** (±0.28)

表6:每个数据集主体类型在不同预测期的预测误差（MASE）。标准差显示在小括号内。注* ：由于滞后期为120的模型长度较短，因此没有对月度数据集进行评估。

主题类型，其中只有一个基于NN的模型超过了先知和默认的NeuralProphet。对人类活动的测量往往具有很强的季节性，并且对当地环境的影响迅速消退。当对遥远的未来进行预测时，统计模式往往主导着人类活动数据。

4.3. 先知和神经先知的任务依赖性强度

在表7中，我们提出了我们的主观建议，即根据手头的任务使用哪个模型。这些是基于我们在评估这两个模型时发现的经验性证据的定性建议。

任务	先知	神经先知
小型数据集（T ″ 100）。	X	
中型到大型数据集（T ″ 100）。		X
长期预测 (h ″ 100)	X	X
中短程预报（1≤h ″ 1000）。		X
具体的预测范围（如h=24）。		X
自动相关（对以前的观察结果的依赖）。		X
滞后的回归者（对观察到的协变量的依赖）。		X
非线性动力学		X
小组数据集的全球建模		X
在小的数据集上频繁地进行再训练，有有限的计算资源	X	
快速预测（计算推理时间）。		X

表7：Prophet和NeuralProphet的任务依赖强度的比较。X标志着我们建议在给定的任务中使用这两个模型中的哪一个。

5. 总结

NeuralProphet是第一个混合预测框架，符合Facebook Prophet设定的可解释性和使用简单性的行业标准。由于NeuralProphet基于PyTorch，并采用标准的深度学习方法进行训练，任何可通过小批量随机梯度下降训练的算法都可以作为一个模块包含在内，这使得开发人员可以很容易地用新功能扩展框架，并采用新的研究。

NeuralProphet可以通过自动回归和协方差回归来模拟本地环境，使其成为一个适合于时间序列的工具，在这种情况下，近期的未来取决于系统的当前状态。我们的实证结果证明了这一点，启用自动回归的NeuralProphet在所有预测范围内都明显优于Prophet。两种模型在配置相同的情况下表现相当。在中短期的预测范围内，NeuralProphet的大多数配置将预测误差降低了50%到90%。将该模型与神经网络相融合，进一步提高了中长期预测的准确性。

在我们的消融研究中，我们观察到NeuralProphet的性能对特定的滞后数或神经网络配置不敏感。

由于坚实的默认和自动超参数，预测初学者可以使用NeuralProphet，而高级用户可以通过可选的模型定制将领域知识纳入其中。使用NeuralProphet升级Prophet后，预测从业者可以在一个方便和可扩展的框架内使用可解释的模型，涵盖广泛的预测应用。

鸣谢

我们感谢Meta、Face-book AI、Netflix、Uber、Monash和Skoltech的学术和工业顾问的宝贵意见；特别是Italo Lima、Caner Komurlu、Alessandro Panella、Evgeny Burnaev、Sean Taylor和Benjamin Letham。我们感谢道达尔能源公司的持续支持；特别是Lluvia Ochoa, Gonzague Henri和Michel Lutz。我们还要感谢我们可爱的开源贡献者的辛勤工作；特别是Mateus De Castro Ribeiro, Riley De Haan, 和Bernhard Hausleitner。

本文介绍的工作部分由斯坦福大学和Total Energies之间的研究协议资助。本文所表达的作者的观点和意见不一定说明或反映资金来源的观点。

参考文献

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017).Stan：一种概率性编程语言。*Journal of Statistical Software*, *76*. doi:10.18637/jss.v076.i01.

Crone, S. F. (2008).Nn5竞争。网址：neural-forecasting-competition.com/NN5。

Doe, N. (2015).Sanfrancisco每小时太阳地面照度数据，单位是w/m2，来自国家太阳辐射 数据库。 URL: nsrdb.nrel.gov/about/tmy.

Dua, D., & Graff, C. (2017).Uci机器学习资源库。URL: archive.ics.uci.edu/ml.

ERCOT （2021）。Ercot控制区2004年1月至2021年9月的每小时负荷，每小时负荷数据档案。URL: ercot.com/gridinfo/load/load_hist.

Girshick, R. (2015).Fast r-cnn. arXiv:1504.08083.

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., & Montero-Manso, P. (2021).莫纳什时间序列 预测档案。arXiv:2105.06643。

Harvey, A. C., & Shephard, N. (1993).结构性时间序列模型。In ***Handbook of Statistics*** (pp. 261-302).Amsterdam:北荷兰卷第11卷：计量经济学。((Edited by G.S. Maddala, C.R. rao and H.D. vinod) ed.).

Hyndman, R. (2018).时间序列预测比赛的简史。URL: robjhyndman.com/hyndsight/ forecasting-competitions。

Hyndman, R., & Athanasopoulos, G. (2014).*预测：原则和实践*。OTexts.URL:books.google.com/books?id=gDuRBAAAQBAJ.

Hyndman, R. J., & Koehler, A. B. (2006).再看一下预测准确性的措施。***International Journal of Forecasting*** , ***22*** , 679-688.URL: https://www.sciencedirect.com/science/article/pii/S0169207006000239. doi:https://doi.org/10.1016/j.ijforecast.2006.03.001.

Loshchilov, I., & Hutter, F. (2019).解耦权重衰减正则化。 arXiv:1711.05101。

Makridakis, S., & Hibon, M. (2000).m3-竞赛：结果、结论和影响。***Int.J. Forecast.***, ***16*** , 451-476.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018).统计和机器学习预测方法。关注的问题和前进的方向。***PLOS ONE*** , ***13*** , e0194889. doi:10.1371/journal.pone.0194889.

Nocedal, J., & Wright, S. (2006).***Numerical optimization***.Springer Science and Business Media.

OpenEI （2015）。圣弗朗西斯科医院每小时的设施耗电量数据，单位：千瓦。URL: data.openei.org/submissions/153.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A. 。Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. （2019）。Pytorch。一个命令式的、高性能的深度学习库。在 H.Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. Fox, & R. Garnett (Eds.), ***Advances in Neural Information Processing Systems 32*** (pp. 8024-8035).Curran Associates, Inc.

RTE （2018）。2017年1月至2018年12月法国公用事业公司RTE的每小时耗电量记录。URL: services-rte.com/en/download-data-published-by-rte.

Smith, L. N. (2017).训练神经网络的循环学习率。 arXiv:1506.01186。

Smith, L. N., & Topin, N. (2018).Super-convergence:使用大的学习 率对神经网络进行非常快速的训练。arXiv:1708.07120。

Svetunkov, I. (2021).用adam进行预测和分析.OpenForecast.URL: https://openforecast.org/adam/.Tashman, L. J. (2000).预测准确性的样本外测试：分析和回顾。***International Journal of Forecasting*** , ***16*** , 437-450.URL: https://www.sciencedirect.com/science/article/pii/S0169207000000650.

doi:https://doi.org/10.1016/S0169-2070(00)00065-0. M3-竞争。

Taylor, S. J., & Letham, B. (2017).规模化的预测。 ***PeerJ*** , .URL: doi.org/10.7287/peerj.preprints.3190v2。 doi:10.7287/peerj.preprints.3190v2。

Triebe, O., Laptev, N., & Rajagopal, R. (2019).Ar-net。 用于时间序列的简单自动回归神经网络。 arXiv:1911.12436.

Uber（ 2018 ）。Omphalos，uber的平行和可扩展语言的时间序列回测工具。网址：eng.uber.com/ omphalos/。

Uber （2020） 。 建立一个回溯测试服务， 以衡量模型在uber规模的表现。 URL: eng.uber.com/ backtesting-at scale/.

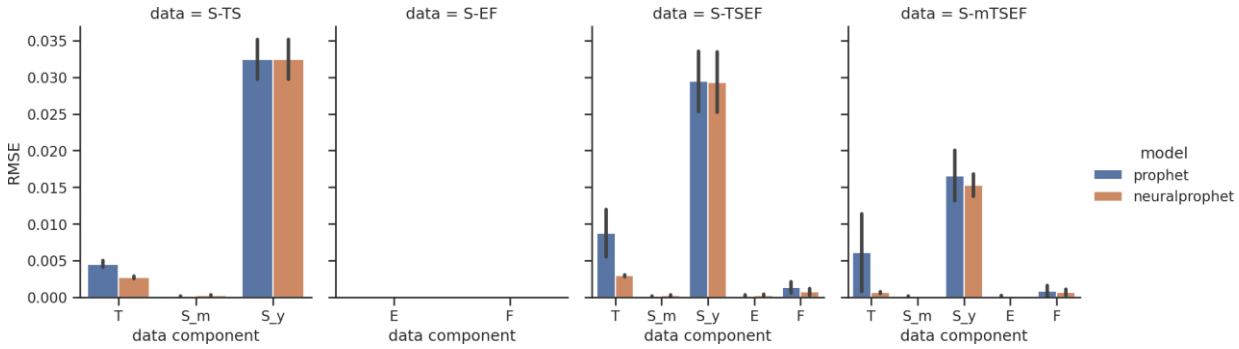
6. 附录

6.1. 附录A：关于数据集的细节

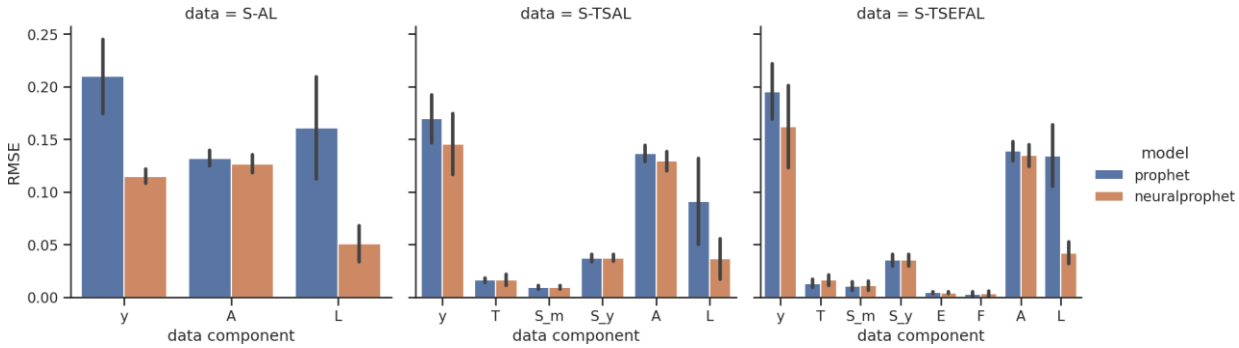
名称	类型	子类型	频率	T	*对数*$_{10}$ *(T)*
空中乘客	人类	旅游	M	144	2.2
零售销售	经济	销售	M	293	2.5
佩顿-曼宁	人类	点击率	D	2905	3.5
谌生	人	人口	D	7305	3.9
负荷医院	能源	负荷	H	8760	3.9
太阳能Sf	环境	太阳	H	8,760	3.9
负载rte	能源	负荷	H	17518	4.2
维多利亚州负载	能源	负荷	30分钟	17520	4.2
yosemite温度	环境	天气	5分钟	18721	4.3
河流	环境	水	D	23741	4.4
价格ercot	经济能源价格	H	25537		4.4
北京的空气	环境	空气质量	H	43800	4.6
太阳黑子	环境	太阳	D	73924	4.9
负载ercot	能源	负荷	H	154872	5.2
风力发电	能源	风力	1分钟	493144	5.7
电力太阳能	能源	太阳能	1分钟	493149	5.7

表8：数据集特征摘要。*T*指的是数据集的长度。

6.2. 附录B：每个实验中每个成分的分解RMSE



(a) 没有滞后成分的实验。



(b) 包括滞后成分的实验。*

图7：这里我们显示每个实验和组件的分解性能。衡量标准是预测的预测成分与基础的真实生成成分值的RMSE。图中的条形显示的是平均值，线显示的是5次运行的标准偏差。*注意：Prophet不支持自动回归和滞后调节器成分。Prophet对上述成分的近似值被隐含地假定为零。因此，Prophet显示的误差等同于成分的标准差。

6.3. 附录C：每个预测范围的RMSSE

	不同预测期的RMSSE			
模型	1步	3步	15步	60步
先知*1		4.37 (±0.99)		
NeuralProphet*1		4.41 (±0.97)		
神经先知 (1滞后)	0.71 (±0.09)	不适用	不适用	不适用
神经先知 (3个滞后)	0.61 (±0.08)	不适用	不适用	不适用
神经先知（12个滞后）。	**0.59** (±0.08)	0.92 (±0.17)	不适用	不适用
神经先知(30滞后)	**0.57** (±0.11)	**0.82** (±0.16)	**1.51** (±0.21)	不适用
神经先知 (120滞后)*2	**0.51** (±0.08)	**0.76** (±0.11)	**1.45** (±0.22)	**2.46** (±0.49)

表9：所有数据集的每个预测范围的预测误差（RMSSE）。标准偏差显示在括号内。注*1 ：无滞后期的模型可以产生任意数量的预测。注*2 ：由于月度数据集的长度较短，没有对具有120个滞后期的模型进行评估。

	不同预测期的RMSSE			
模型	1个步骤	3个步骤	15个步骤	60步
(30个滞后，1x32 NN)	0.51 (±0.07)	0.76 (±0.11)	1.42 (±0.20)	不适用
(30滞后，2x24 NN)	**0.50** (±0.07)	0.75 (±0.11)	1.43 (±0.22)	不适用
(30个滞后，4x16 NN)	0.51 (±0.08)	0.76 (±0.11)	1.44 (±0.21)	不适用
(120滞后，1x32 NN)	0.52 (±0.09)	0.75 (±0.11)	**1.36** (±0.20)	**2.07** (±0.35)
(120滞后，2x24 NN)	0.55 (±0.11)	**0.74** (±0.11)	1.37 (±0.19)	2.09 (±0.39)
(120滞后，4x16 NN)	0.55 (±0.10)	0.78 (±0.12)	1.41 (±0.25)	2.09 (±0.35)

表10：NeuralProphet（含NN）对所有数据集的每个预测期的预测误差（RMSSE），月度除外，因为它们的长度不足以适应NN。标准差显示在括号内。模型定义显示了输入滞后数和层数以及它们的隐藏大小。例如，4x16表示4个隐藏层，维度为16。