南京大学 人工智能学院
SCHOOL OF ARTIFICIAL INTELLIGENCE, NANJING UNIVERSITY

9 时间序列分析的深度学习方法

基础深度学习模型与方法

神经网络预测

ELSEVIER

International Journal of Forecasting

Volume 14, Issue 1, 1 March 1998, Pages 35-62

Forecasting with artificial neural networks:: The state of the art

Guoqiang Zhang, B. Eddy Patuwo, Michael Y. Hu

Journal of FORECASTING

Research article | Full Access

How effective are neural networks at forecasting and prediction? A review and evaluation

Monica Adya, Fred Collopy

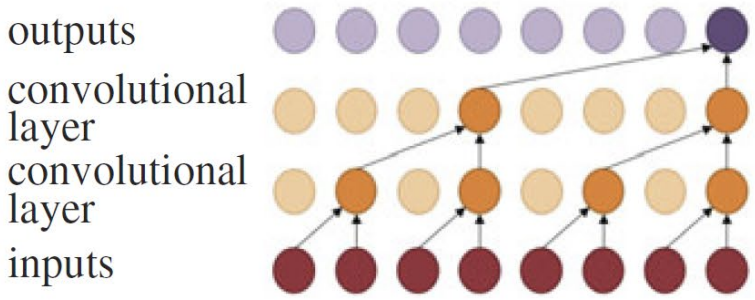
First published: 04 December 1998 |
https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<481::AID-FOR709>3.0.CO;2-Q | Citations: 212

- 1969 Weather forecasting with adaptive linear neurons (Hu)
- 1986 Backpropagation (Rumelhart et al.)
- 1988 NNs using backpropagation applied to forecasting; positive results (Werbos)
- 199x Many authors applying mostly feed-forward models to various forecasting problem (single time series)
- 1998 Review articles: "The outcome of all of these studies has been somewhat mixed"; "While ANNs provide a great deal of promise, they also embody much uncertainty."
- 2000 M3 competition – simple methods declared the winner
- 200x Less work on NN-based forecasting methods
- 2012 AlexNet wins ImageNet competition – start of the Deep Learning revival (Krizhevsky et al.)
- 2014 Generating Sequences With RNNs (Graves); seq2seq architecture (Sutskever et al.)
- 2014- Modern deep learning techniques (RNNs, CNNs) get applied to forecasting (across time series)
- 2018 M4 competition: combination of NNs and classical techniques wins

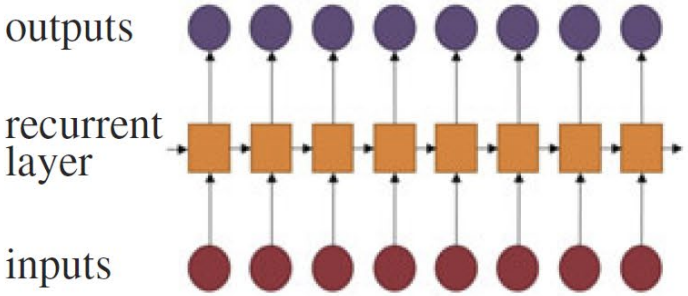
[Jan et al. 2021]

深度学习实现时间序列分析映射

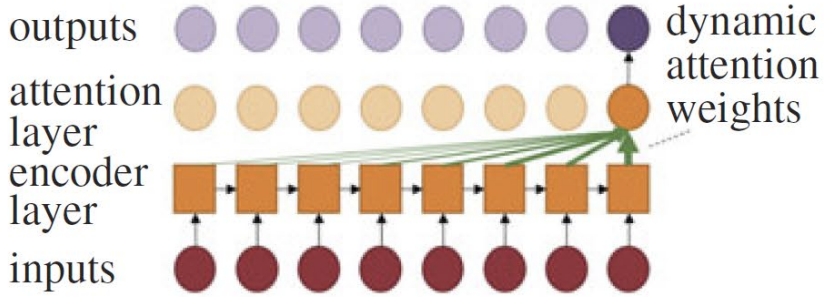
$$\hat{y}_{n+1} = f\left(\boldsymbol{x}_{n+1};\ \begin{matrix}\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n \\ y_1, y_2, \ldots, y_n\end{matrix}\right)$$



CNN model



RNN model



attention-based model

概要

1. 序列模型：
RNN，LSTM

2. 卷积模型：
TCN

3. 注意力模型：
Transformer

4. 其他深度结构：
N-Beats

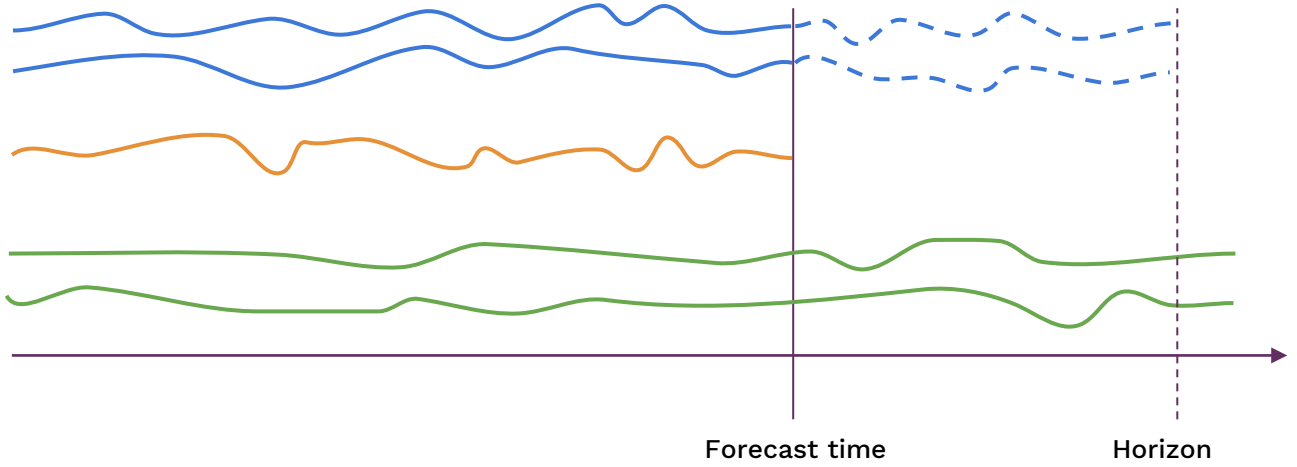
如何**融合统计归纳偏好**?
(Hybrid Model)

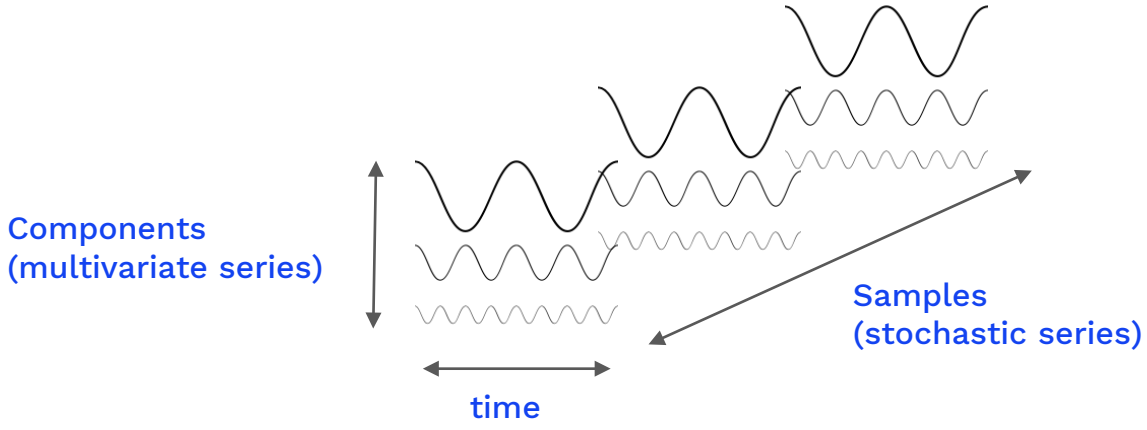
时间序列预测

Target series: what we want to forecast

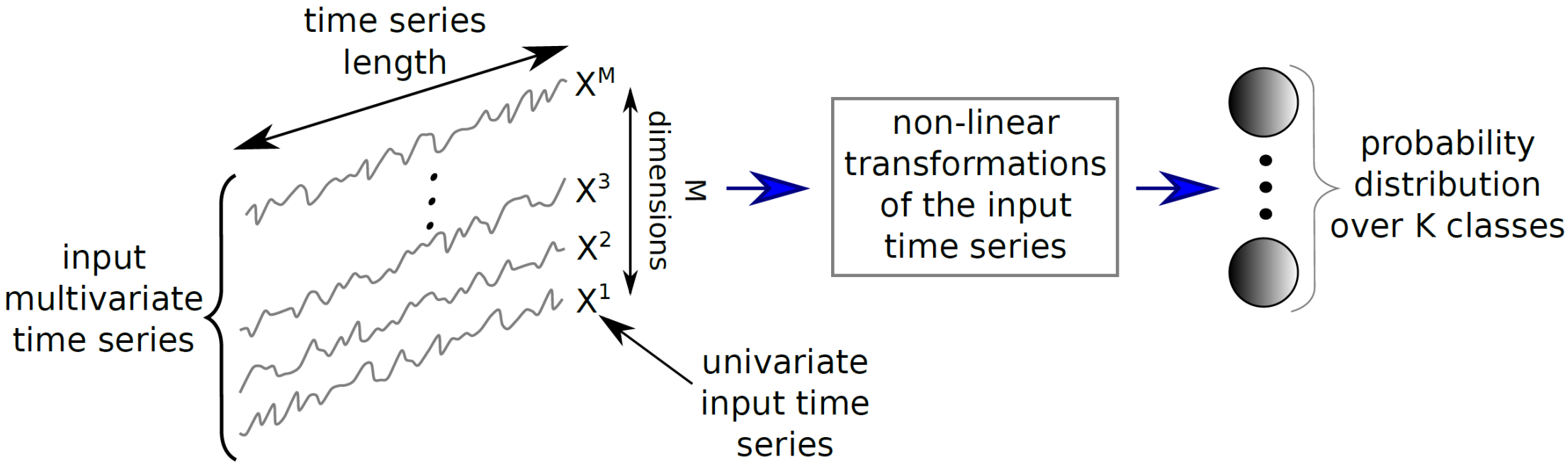
Past covariates: Unknown into the future e.g. measurements

Future covariates: Known into the future e.g. calendar, weather forecasts, actions taken



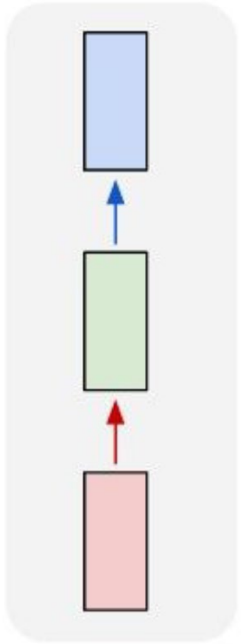


时间序列分类

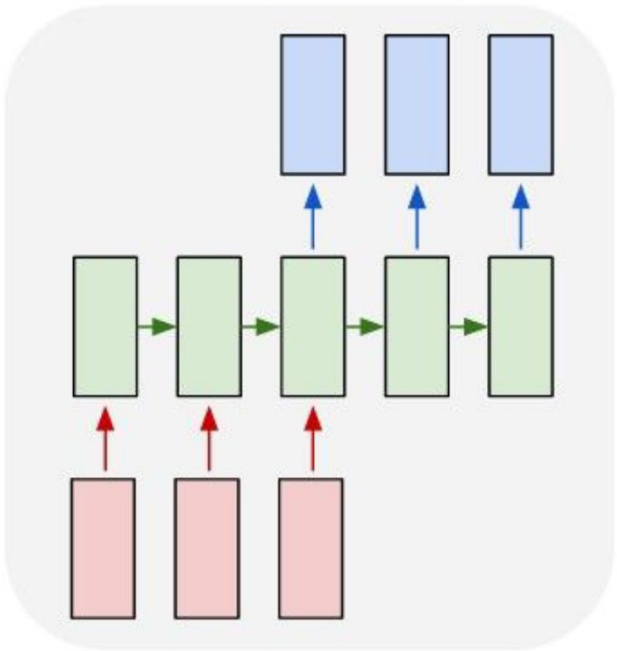


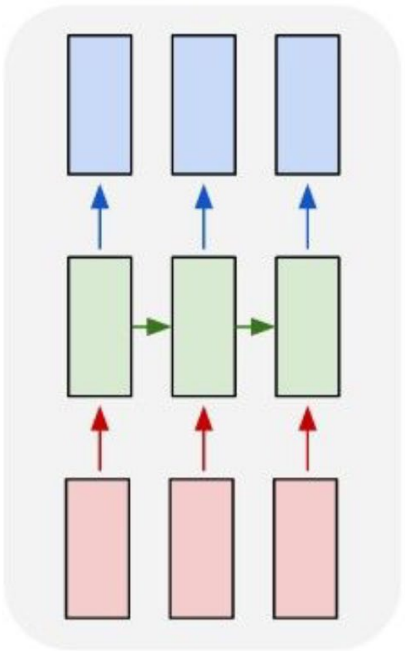
Ismail Fawaz, H., Forestier, G., Weber, J. et al. **Deep learning for time series classification: a review**. Data Mining and Knowledge Discovery. 2019.

序列化模型



一对一模型





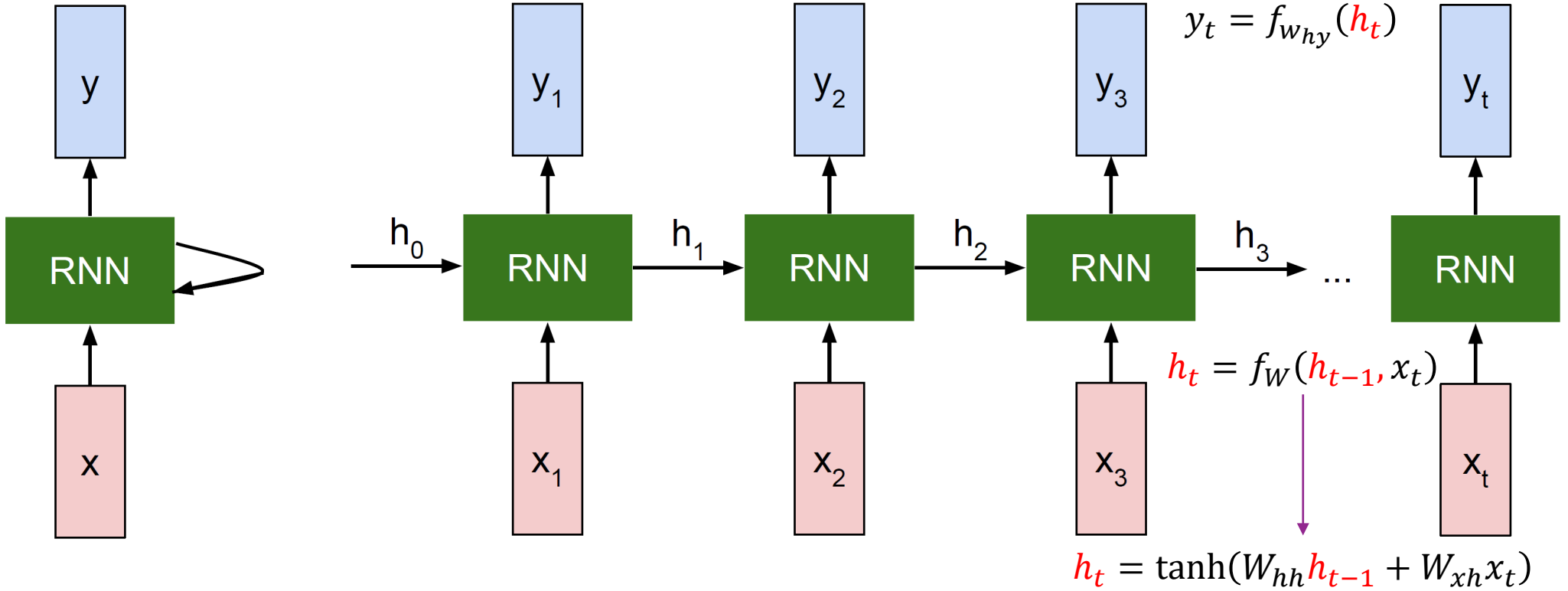
多对多序列模型（如用于视频处理）

- 时序模型（序列化模型）的特点
  - 输入序列长度不定
  - 输出结果长度不定

http://cs231n.stanford.edu/schedule.html

RNN（Recurrent Neural Network）

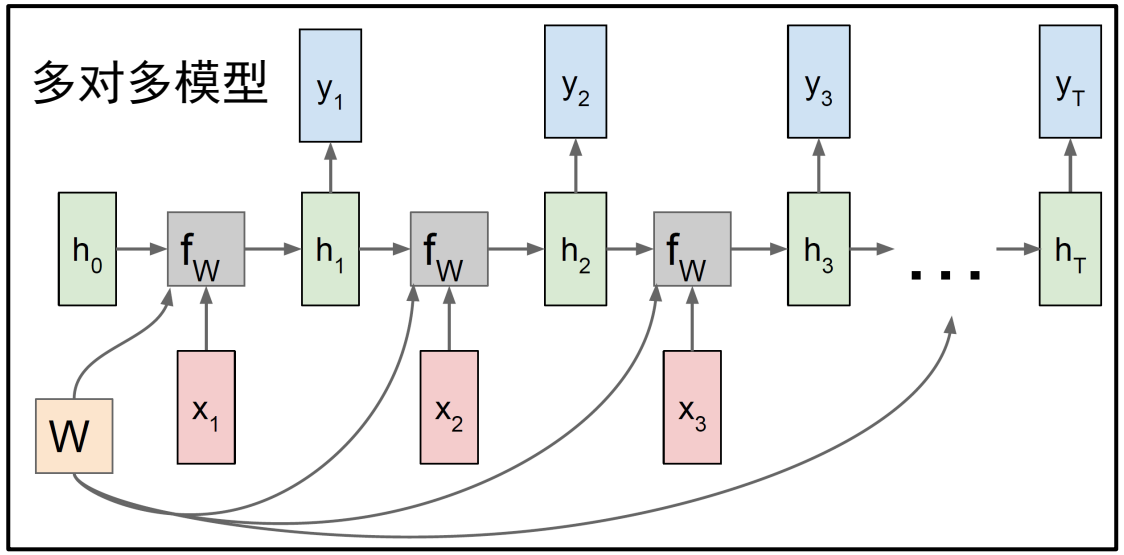
- RNN：使用隐藏状态h_t"整合"历史信息，隐藏状态随时序更新
  - 在不同时间单位之间状态更新/预测参数W是**共享**的

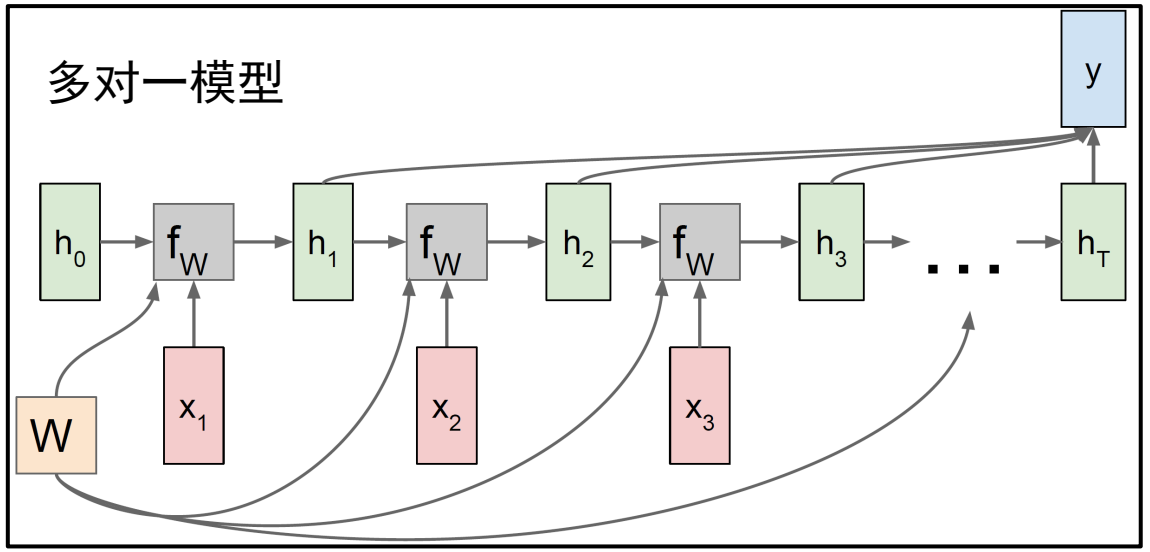


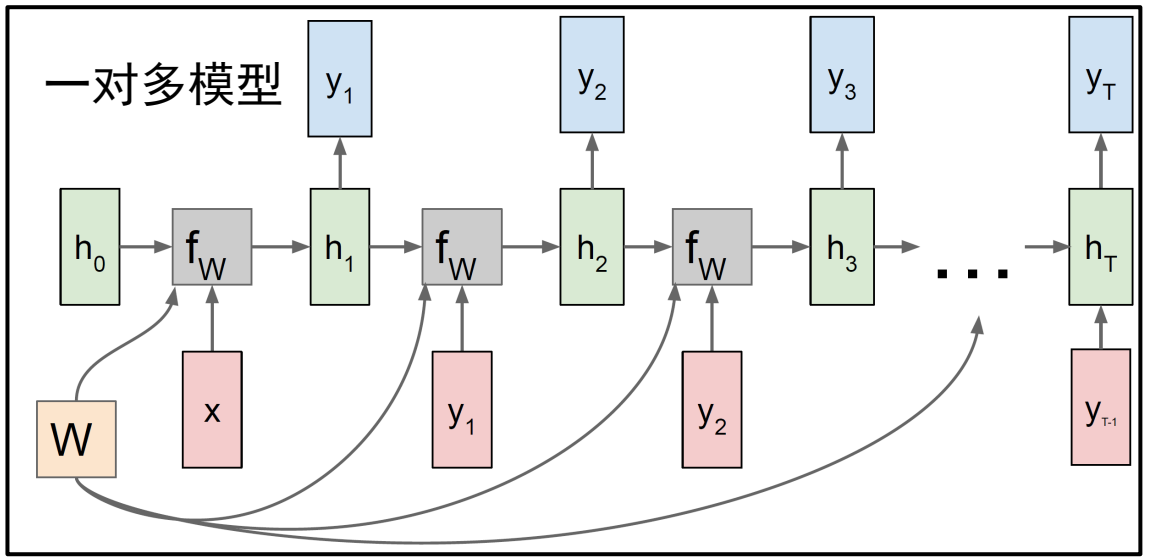
RNN简化表示

RNN展开（unrolled）

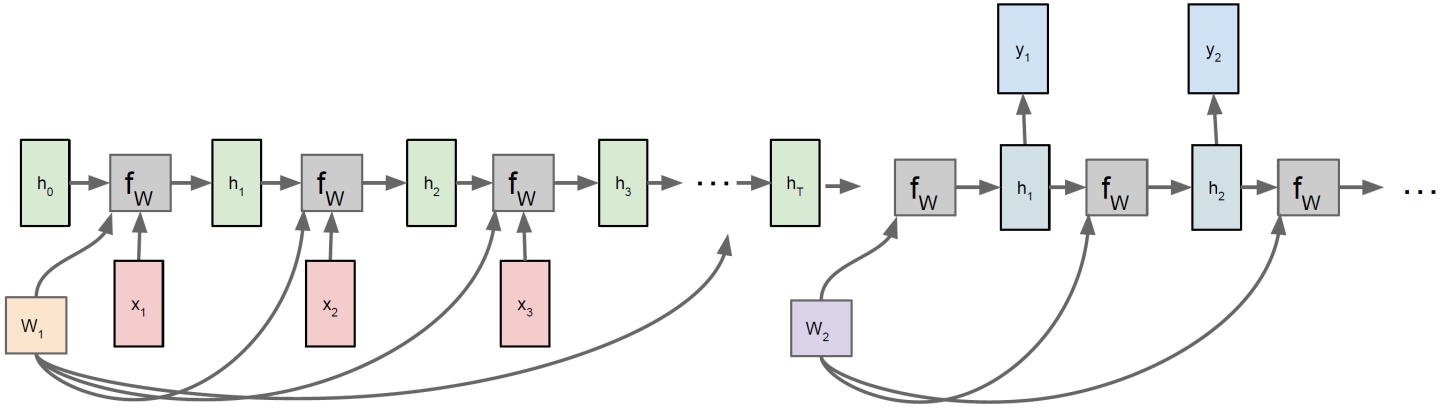
RNN处理不同类型任务

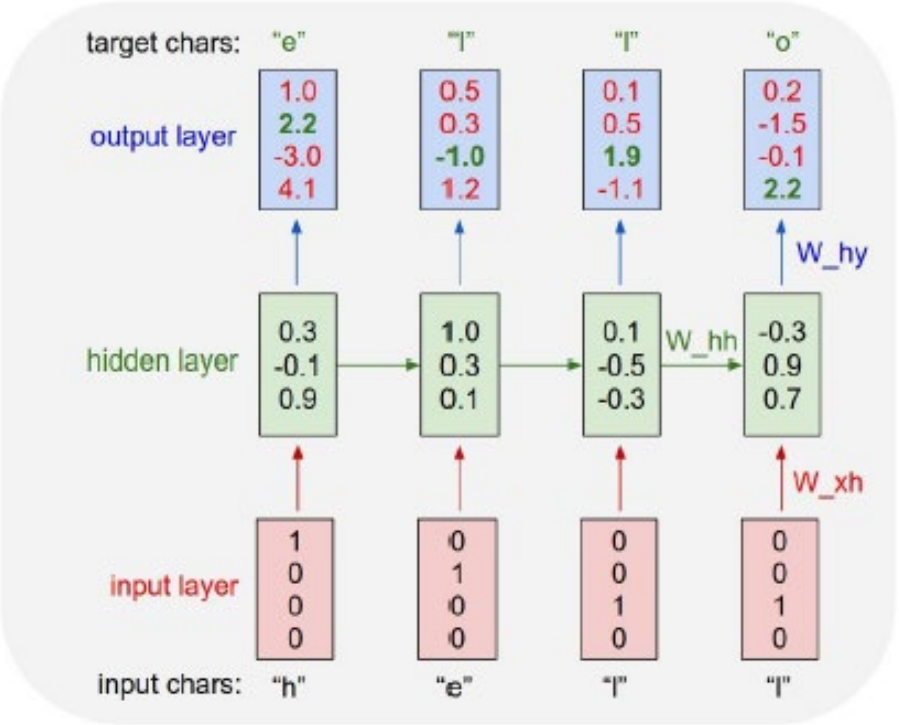






RNN用于序列化预测

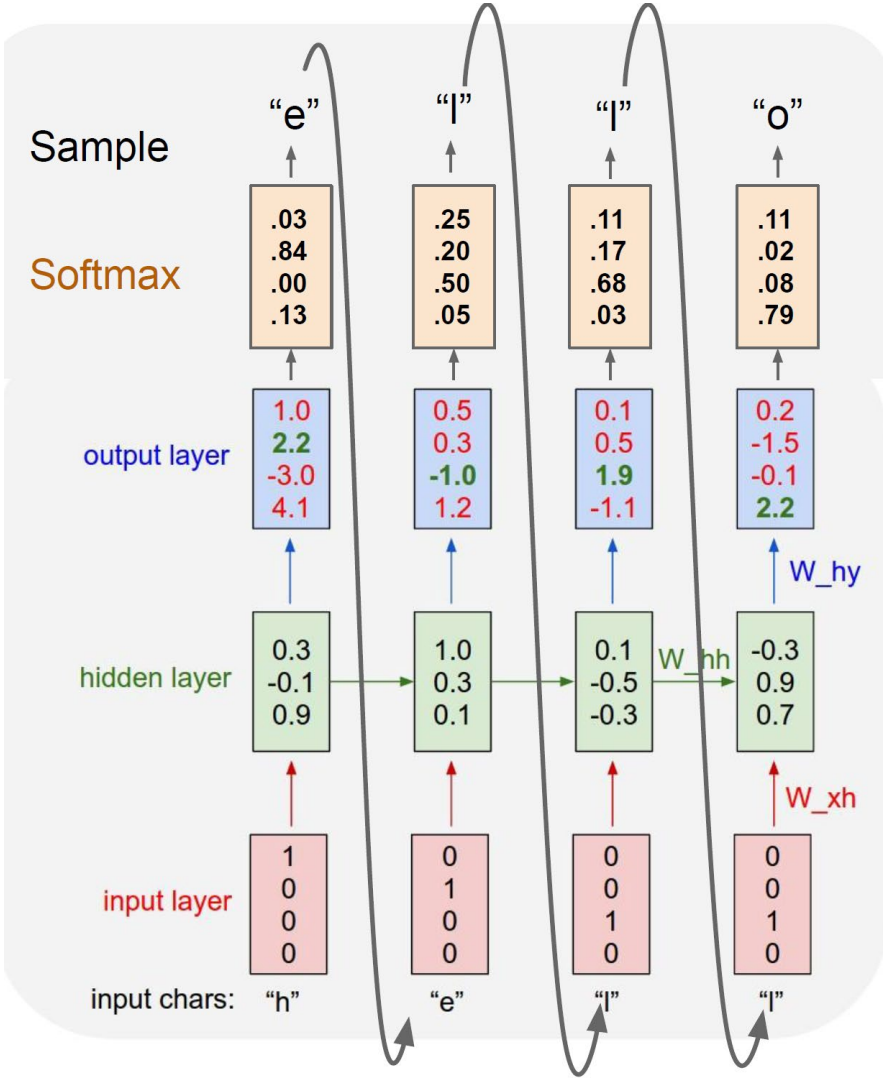




训练阶段

Ilya Sutskever, Oriol Vinyals, Quoc V. Le. **Sequence to Sequence Learning with Neural Networks**. NIPS 2014

RNN用于序列化预测



测试阶段

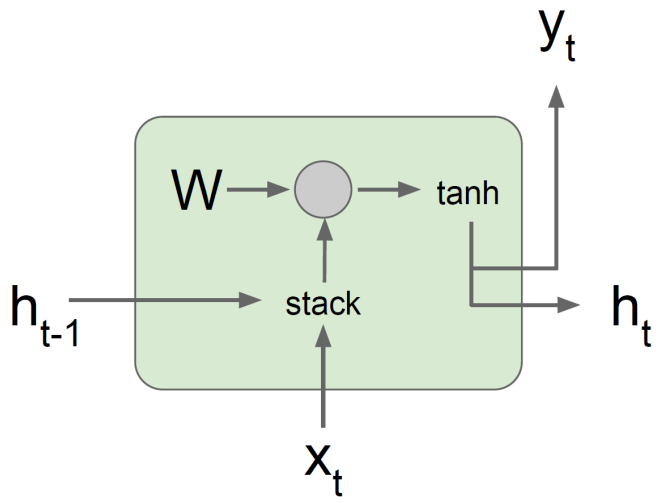
- RNN优点
  - 能够处理变长输入
  - （理论上）能够利用（足够长的）历史信息
  - 模型大小不随序列长度变化
- RNN缺点
  - 计算较慢
  - 实际应用中很难获取**较早的**历史信息

Bengio et al. **Learning long-term dependencies with gradient descent is difficult**. IEEE Transactions on Neural Networks, 1994

Pascanu et al. **On the difficulty of training recurrent neural networks**, ICML 2013

RNN的参数更新

- Forward

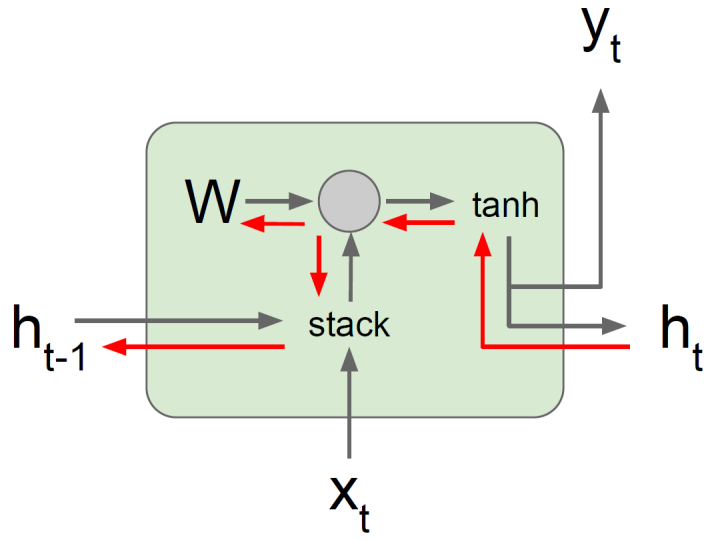


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

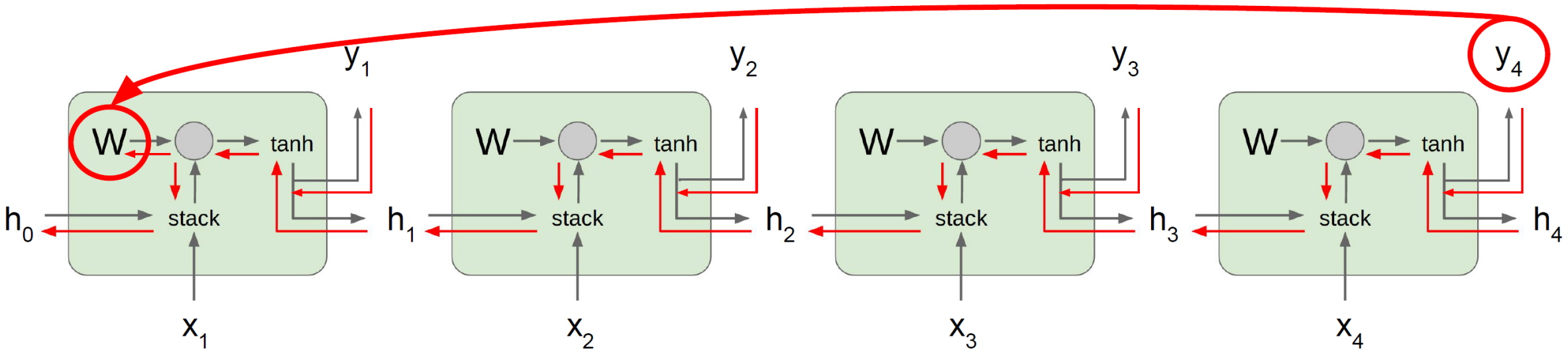
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

- Backward



$$\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}$$

RNN的参数更新



$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T}\frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T}\left(\prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}}\right)\frac{\partial h_1}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T}\left(\prod_{t=2}^{T} \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)\right) W_{hh}^{T-1}\frac{\partial h_1}{\partial W}$$

一般小于1，梯度消失

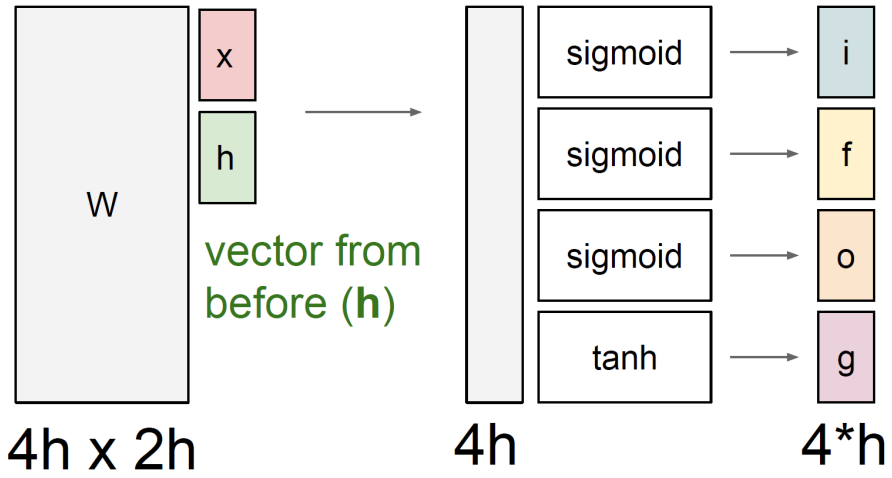
LSTM（Long Short-Term Memory）

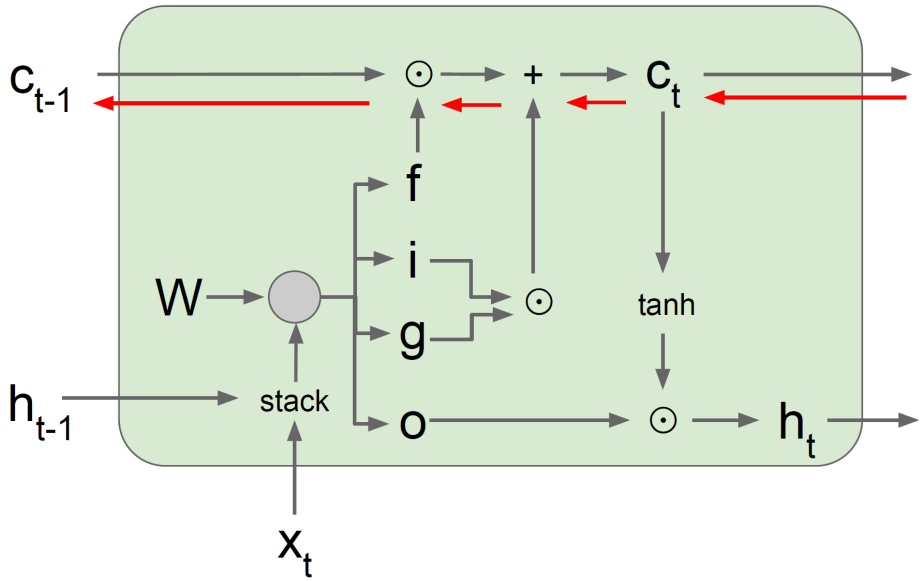
- Gate gate: How much to write to cell
- Input gate: whether to write to cell
- Forget gate: Whether to erase cell
- Output gate: How much to reveal cell

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot tanh(c_t)$$





BP从c_t到c_{t-1}只包含f相关项，不包含矩阵W

Sepp Hochreiter, Jürgen Schmidhuber. **Long Short-Term Memory**. Neural Comput. 9(8): 1735-1780 (1997)

DeepAR

- Produce accurate ***probabilistic*** forecasts, based on training an **auto-regressive recurrent network** model on a large number of related time series.
- Assume the model distribution $Q_\Theta(\mathbf{z}_{i,t_0:T} \mid \mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T})$ as

$$Q_\Theta(\mathbf{z}_{i,t_0:T} \mid \mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}) = \prod_{t=t_0}^{T} Q_\Theta(z_{i,t} \mid \mathbf{z}_{i,1:t-1}, \mathbf{x}_{i,1:T}) = \prod_{t=t_0}^{T} \ell\left(z_{i,t} \mid \theta(\mathbf{h}_{i,t}, \Theta)\right)$$

- Gaussian likelihood for real-valued data

$$\ell_{\mathrm{G}}(z \mid \mu, \sigma) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-(z-\mu)^2/(2\sigma^2)\right)$$
$$\mu(\mathbf{h}_{i,t}) = \mathbf{w}_\mu^T \mathbf{h}_{i,t} + b_\mu \text{ and } \sigma(\mathbf{h}_{i,t}) = \log\left(1 + \exp\left(\mathbf{w}_\sigma^T \mathbf{h}_{i,t} + b_\sigma\right)\right).$$

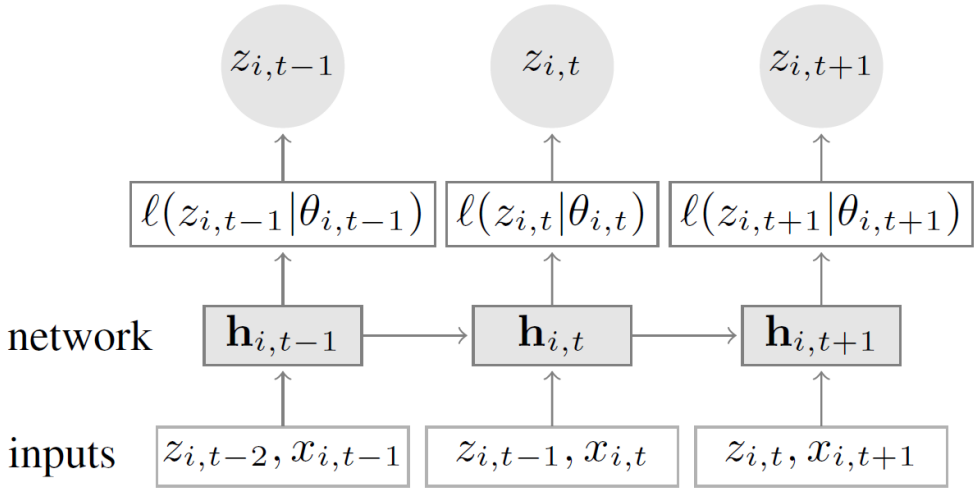
- Negative-binomial likelihood for positive count data

$$\ell_{\mathrm{NB}}(z \mid \mu, \alpha) = \frac{\Gamma\left(z + \frac{1}{\alpha}\right)}{\Gamma(z+1)\Gamma\left(\frac{1}{\alpha}\right)} \left(\frac{1}{1+\alpha\mu}\right)^{\frac{1}{\alpha}} \left(\frac{\alpha\mu}{1+\alpha\mu}\right)^{z}$$
$$\mu(\mathbf{h}_{i,t}) = \log\left(1 + \exp\left(\mathbf{w}_\mu^T \mathbf{h}_{i,t} + b_\mu\right)\right) \text{ and } \alpha(\mathbf{h}_{i,t}) = \log\left(1 + \exp\left(\mathbf{w}_\alpha^T \mathbf{h}_{i,t} + b_\alpha\right)\right)$$

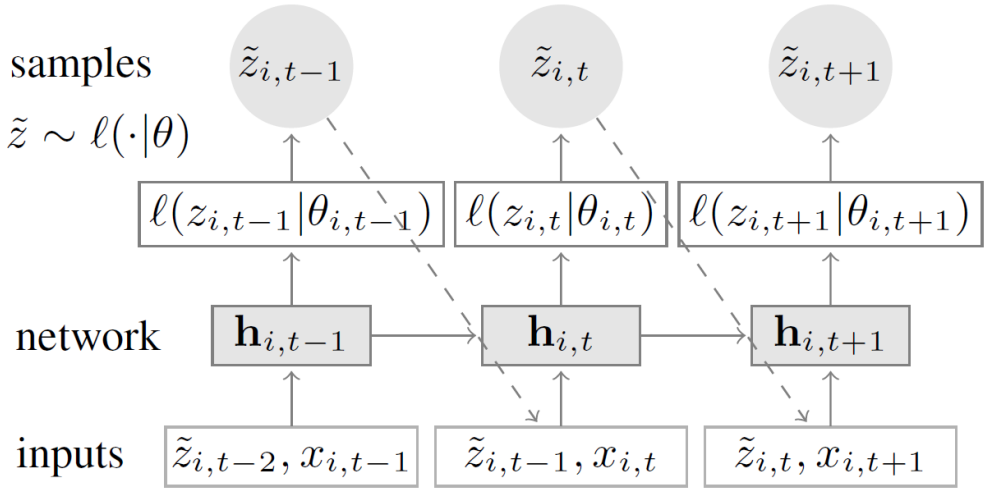
Valentin Flunkert, David Salinas, Jan Gasthaus. **DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks**. CoRR abs/1704.04110 (2017)

DeepAR

- Produce accurate ***probabilistic*** forecasts, based on training an **auto-regressive recurrent network** model on a large number of related time series.



Training



Prediction

The network output $\boldsymbol{h}_{i,t} = h(\boldsymbol{h}_{i,t-1}, z_{i,t-1}, \boldsymbol{x}_{i,t}, \Theta)$ is then used to compute the parameters $\theta_{i,t} = \theta(\boldsymbol{h}_{i,t}, \Theta)$ of the likelihood $\ell(z \mid \theta)$.

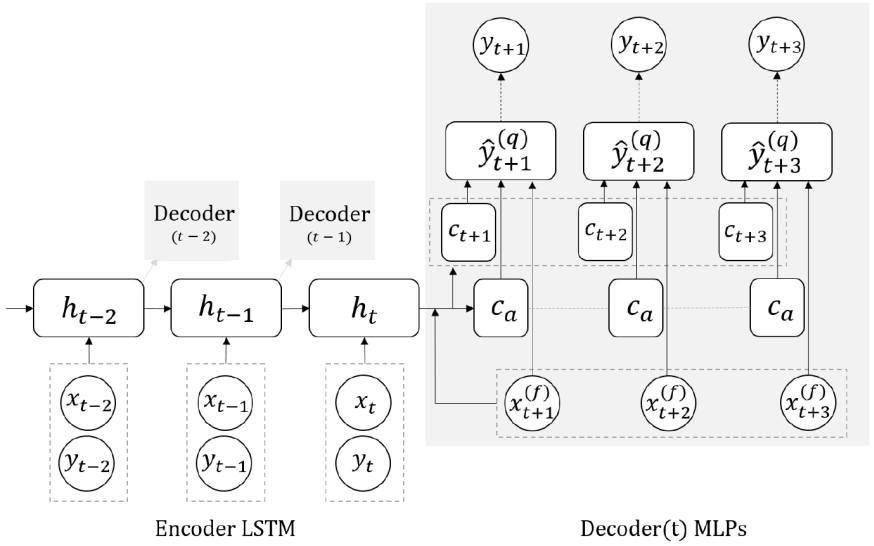
the history of $z_{i,t}$ is fed in for $t < t_0$, a sample $\hat{z}_{i,t} \sim \ell(\cdot \mid \theta_{i,t})$ is drawn and fed back for the next point until the end of the prediction range $t = t_0 + T$

MQ-RNN

- Minimize the total Quantile Loss (QL)

$$\min \sum_t \sum_q \sum_k L_q \left(y_{t+k}, \hat{y}_{t+k}^{(q)} \right)$$

$$L_q(y, \hat{y}) = q(y - \hat{y})_+ + (1 - q)(\hat{y} - y)_+$$

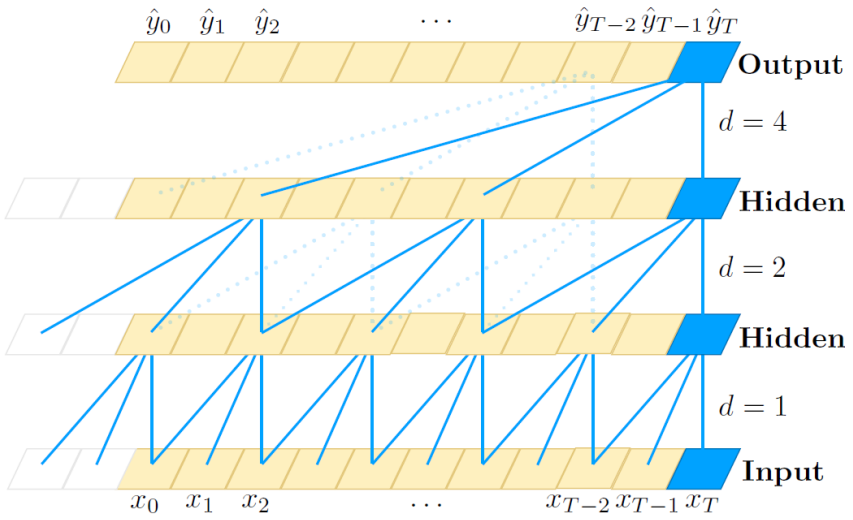


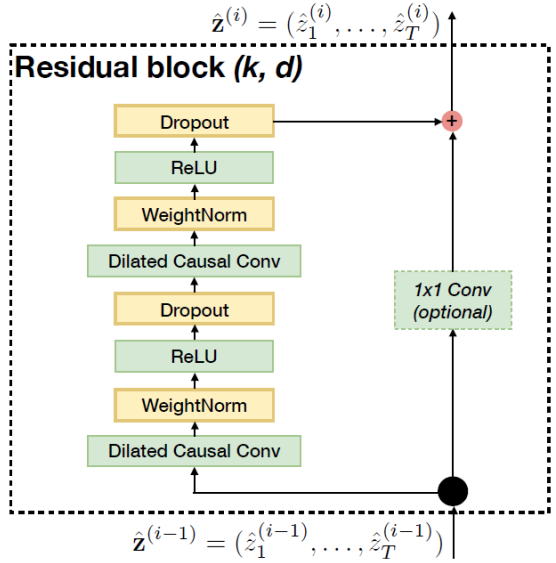
- MQ-RNN has a design of **two MLP branches**.
- The first (global) MLP summarize the encoder output plus all future inputs into two contexts:
  - a series of horizon-specific contexts c_{t+k} for each of the K future points,
  - and a horizon-agnostic context c_a which captures common information:
- The second (local) MLP applies to each specific horizon. It combines the corresponding future input and the two contexts from the global MLP described earlier, then outputs all the required quantiles for that specific future time step.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy. **A Multi-Horizon Quantile Recurrent Forecaster**. CoRR 2017.

Temporal Convolutional Networks (TCN)

- RNN or CNN?
- Temporal Convolutional Networks
  - 因果卷积（Casual Convolution）
  - 变长输入，固定输出：1D卷积
  - 膨胀卷积（Dilated Convolutions）
  - Residual Connections





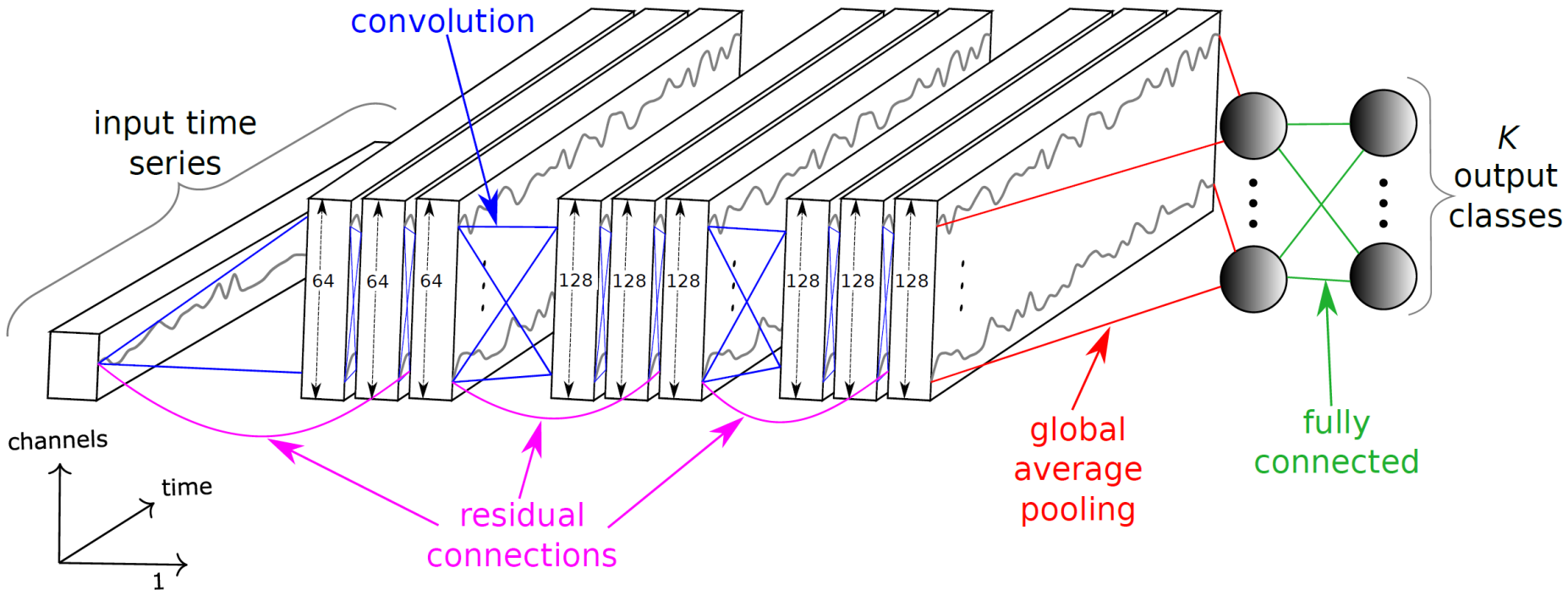
- 主要优势
  - 并行化（同时处理所有时间节点）
  - 可变receptive field
  - 低内存占用
- 缺点（相比RNN）
  - 测试阶段需要整个历史数据
  - 任务迁移性变弱（不同任务对历史数据需求变化）

Shaojie Bai, J. Zico Kolter, Vladlen Koltun. **An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling**. CoRR 2018.

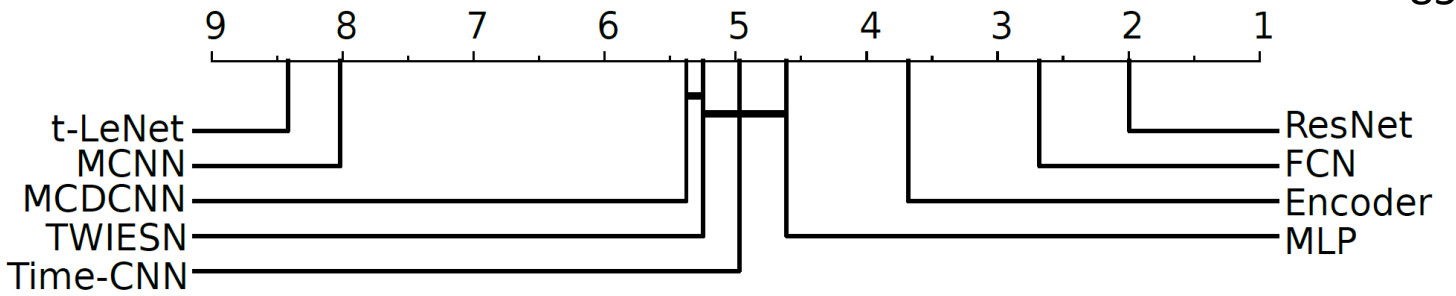
Temporal Convolutional Networks (TCN)

Sequence Modeling Task	Model Size ($\approx$)	Models			
		LSTM	GRU	RNN	**TCN**
Seq. MNIST (accuracy^h)	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^\ell$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^\ell$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^\ell$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

TCN用于时间序列分类

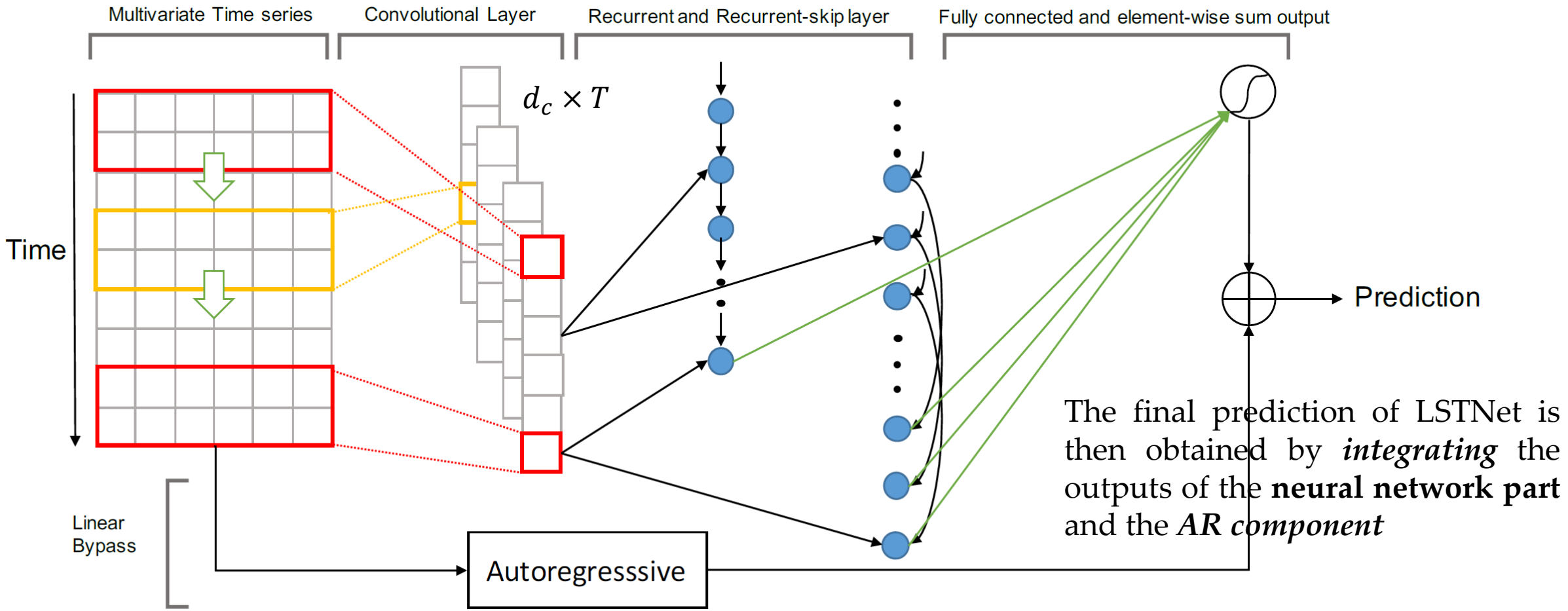


85 problems in total.



Long- and Short-term Time-series network (LSTNet)

Conv: extract ***short-term*** patterns as well as ***local*** dependencies between variables

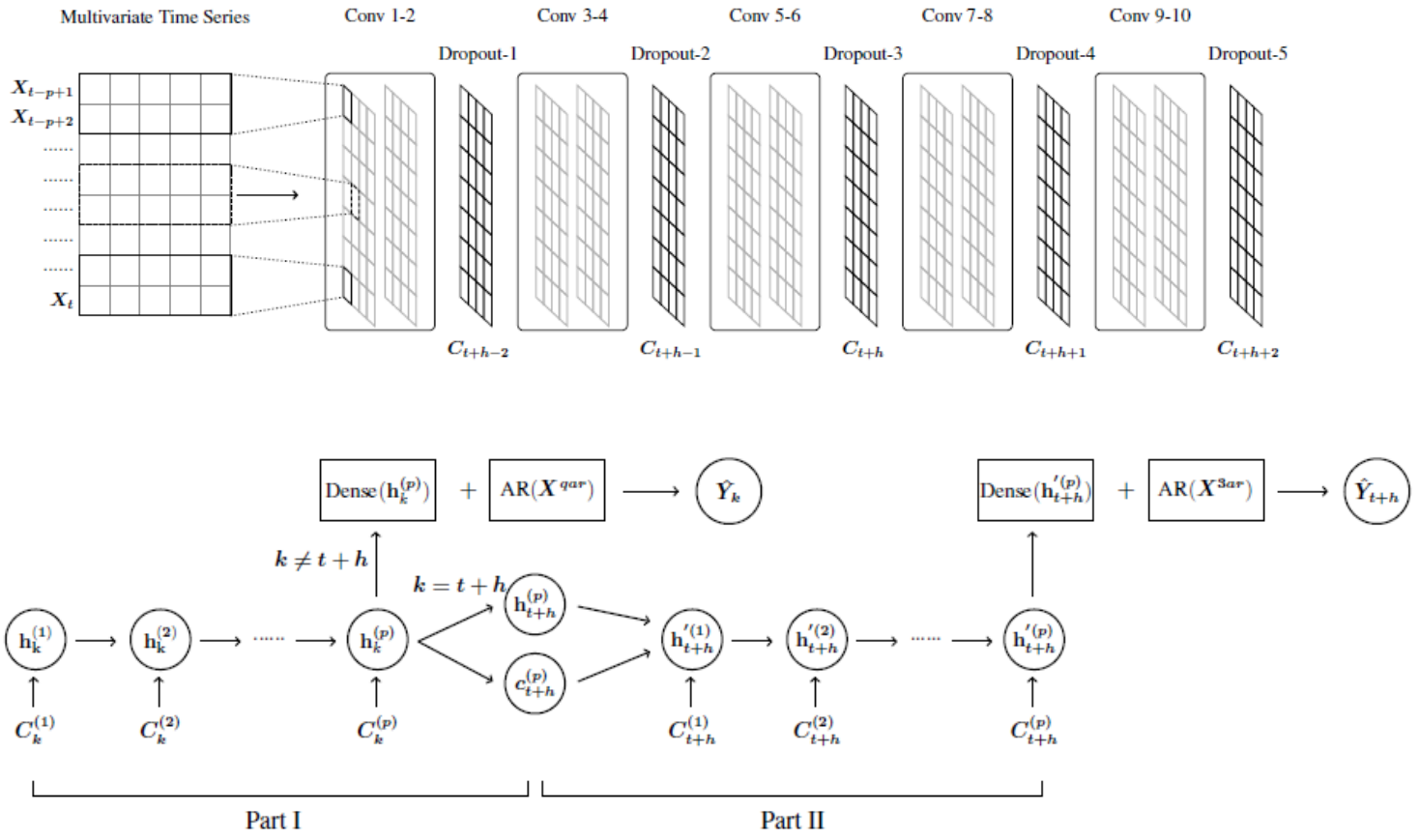


The final prediction of LSTNet is then obtained by ***integrating*** the outputs of the **neural network part** and the ***AR component***

Guokun Lai, Wei-Cheng Chang, Yiming Yang, Hanxiao Liu. **Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks**. SIGIR 2018: 95-104

Multi-Level Construal Neural Network (MLCNN)

- Use CNN to extract multi-level abstract representations of the raw data *for near and distant future predictions*.
- Model the **interplay** between multiple predictive tasks and fuse their future visions through a modified Encoder-Decoder architecture.
- Combine traditional AR model with the neural network to solve the scale insensitive problem.

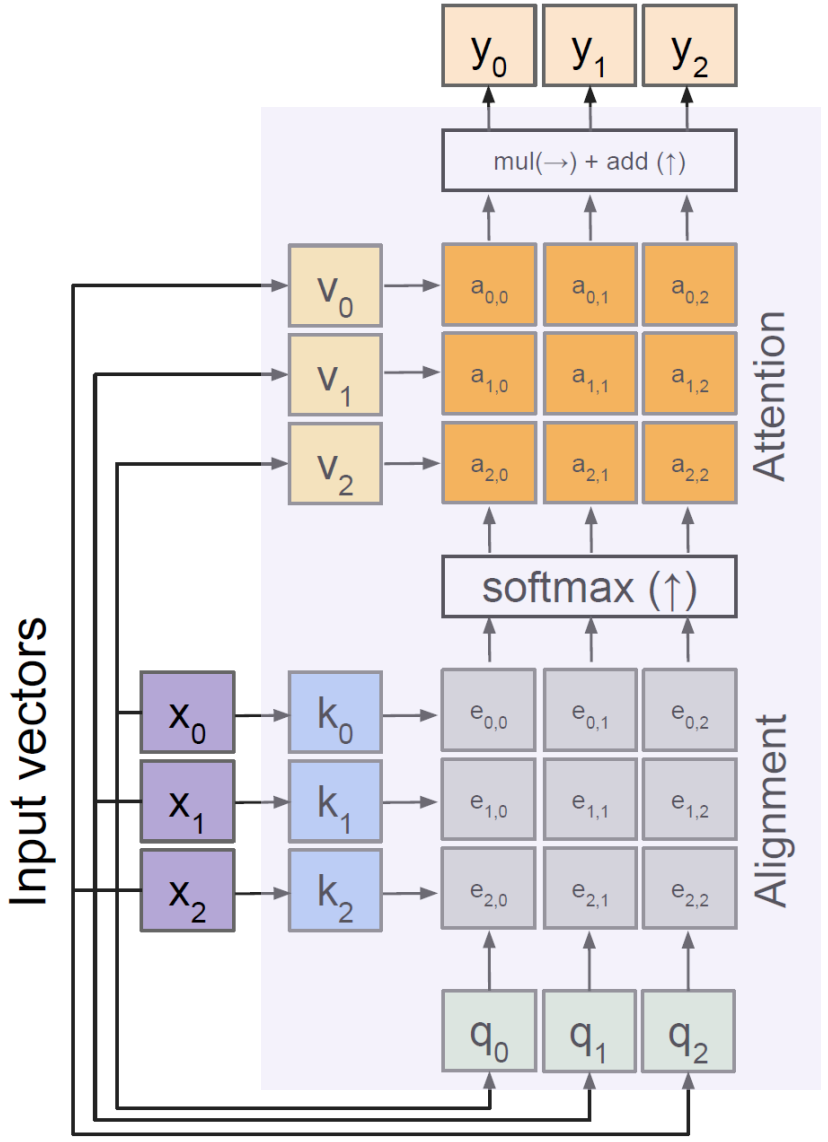


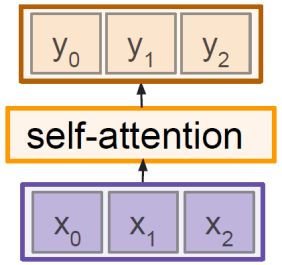
Part I: The shared LSTM for all tasks;
Part II: The main LSTM for main task

Jiezhu Cheng, Kaizhu Huang, Zibin Zheng. **Towards Better Forecasting by Fusing Near and Distant Future Visions**. AAAI 2020: 3593-3600

基于注意力的时序建模

- 自注意力（Self-Attention），用于处理针对集合的特征变换
- 输入：$\boldsymbol{x} \in \mathbb{R}^{N \times D}$
- 输出：$\boldsymbol{y} \in \mathbb{R}^{N \times D_v}$
- 自注意力
  - Query: $\boldsymbol{q} = \boldsymbol{x}\boldsymbol{W}_q$;
  - Key: $\boldsymbol{k} = \boldsymbol{x}\boldsymbol{W}_k$; Value: $\boldsymbol{v} = \boldsymbol{x}\boldsymbol{W}_v$
  - $e_{ij} = q_j \cdot k_i / \sqrt{D}$
  - $\boldsymbol{a} = \mathbf{softmax}(\boldsymbol{e})$
  - $y_j = \sum_i a_{ij} v_i$





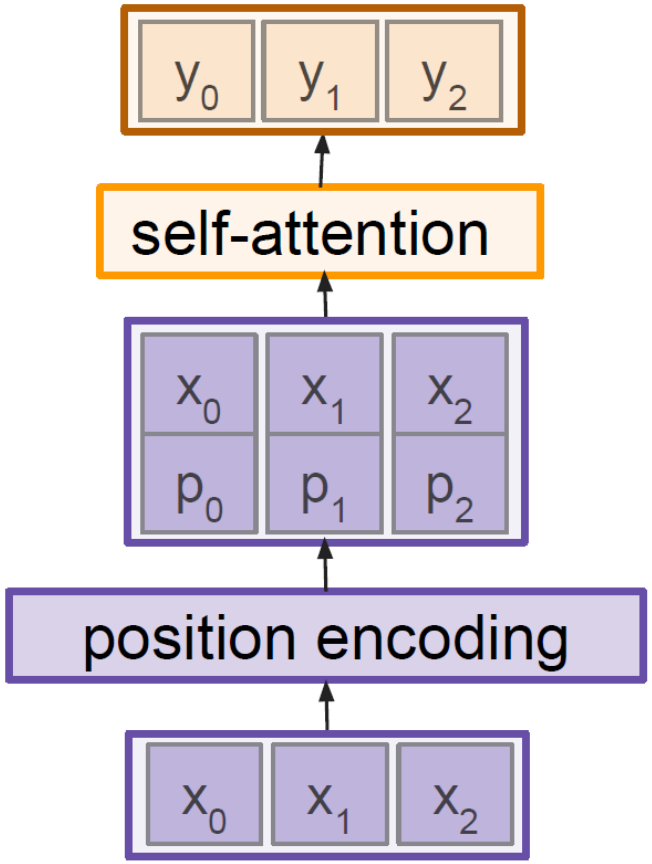
Ashish Vaswani et al. Attention is All you Need. NIPS 2017: 5998-6008

http://cs231n.stanford.edu/slides/2021/lecture_11.pdf

位置编码

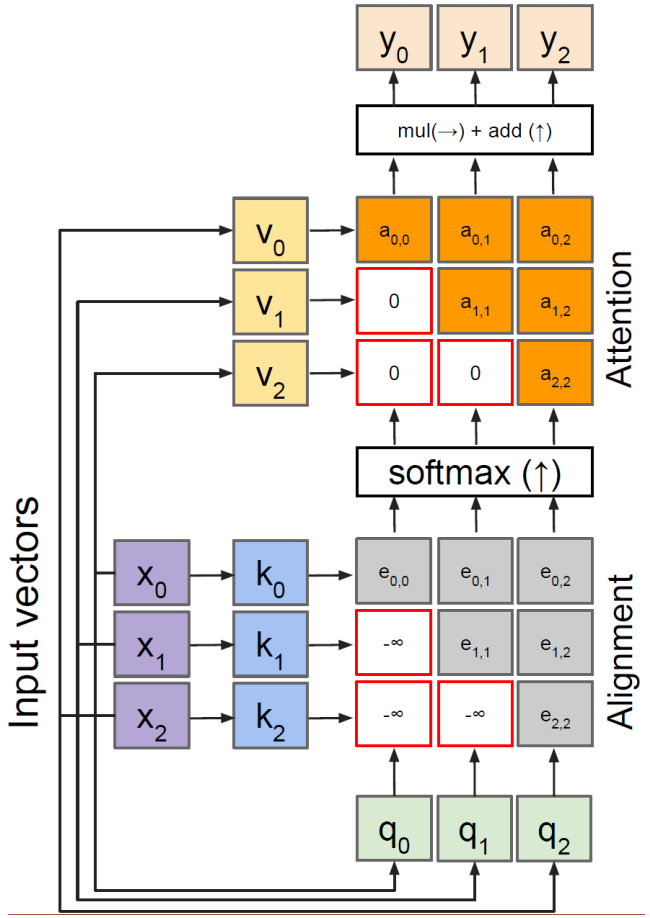
- 使用位置编码（Positional Encoding）表示输入的位置信息
- 定义位置映射 $pos: N \to \mathbb{R}^d$，有$p_j = pos(j)$
  - 不同位置有唯一的位置编码
  - 位置编码的距离能够反映实际位置的远近
  - 能够扩展到更长的序列中
- 选项1：学习不同位置的编码字典，位置编码从对应**位置**查询
- 选项2：定义位置映射

$$\mathbf{p}(t) = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_d \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

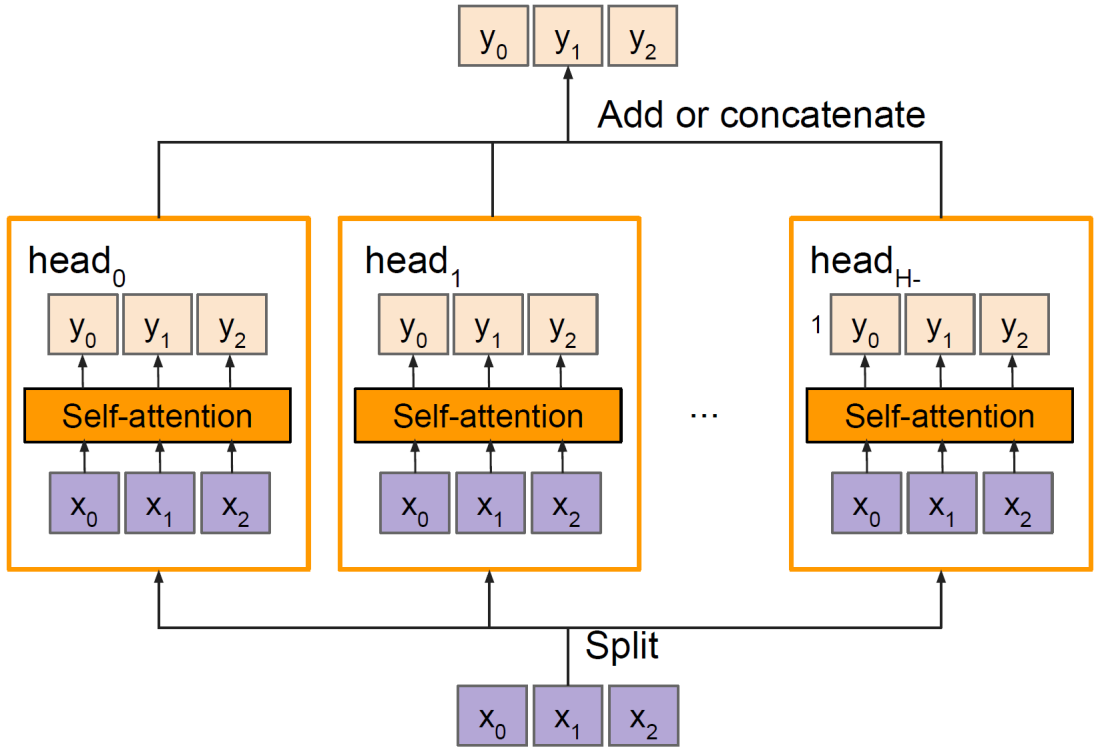


自注意力的推广

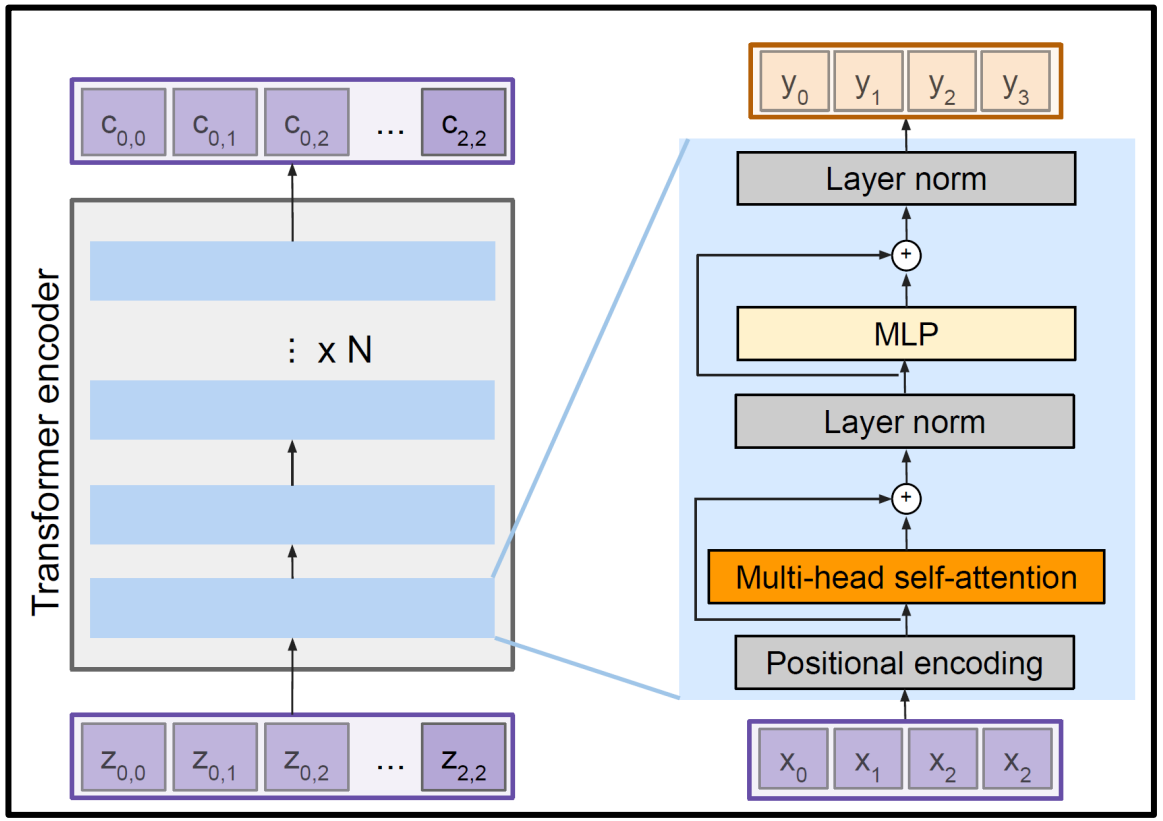
- 具有掩码的Attention
  - 设置未见值为−∞

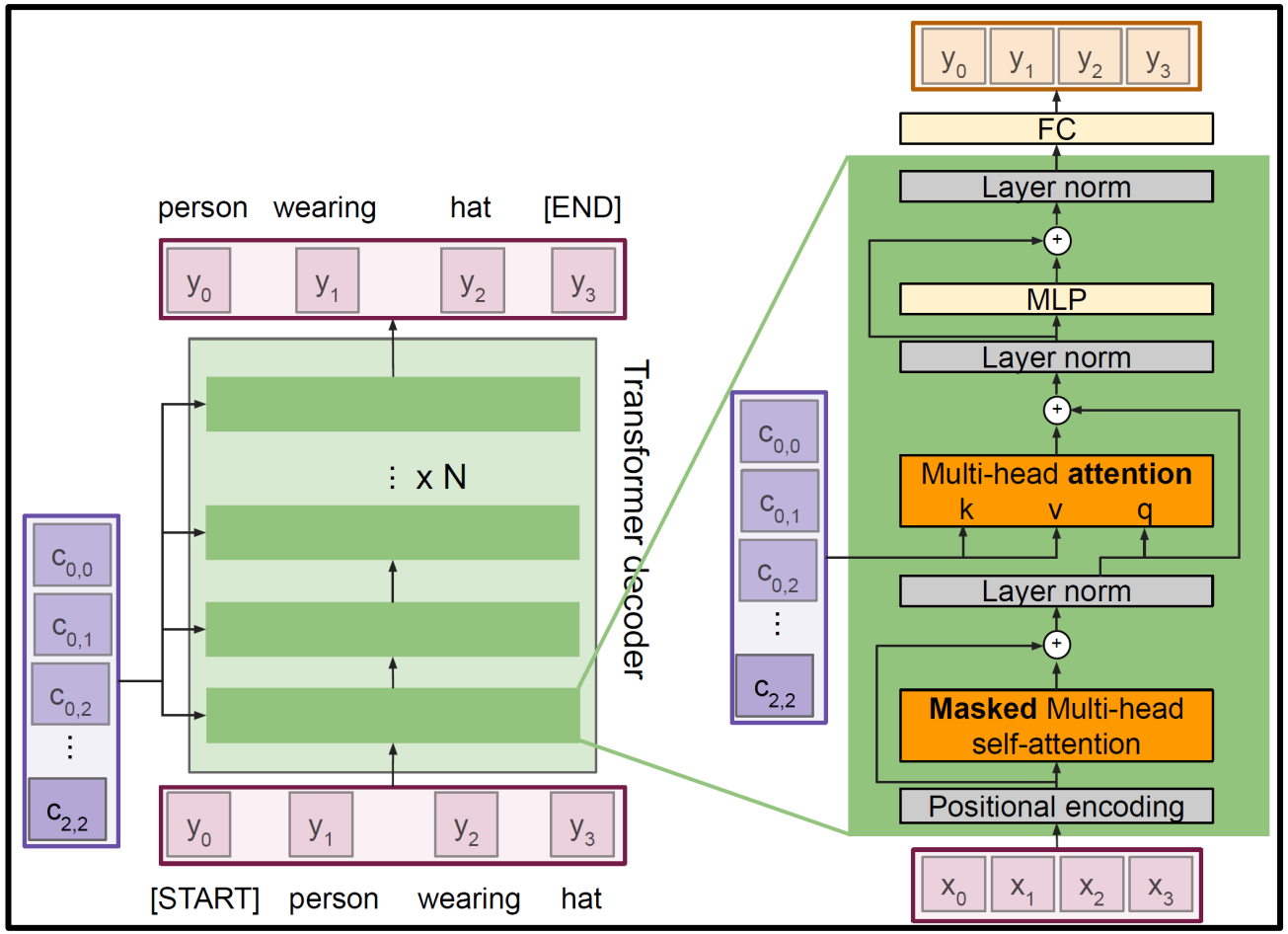


- Multi-Head/Multi-Layer



Transformer

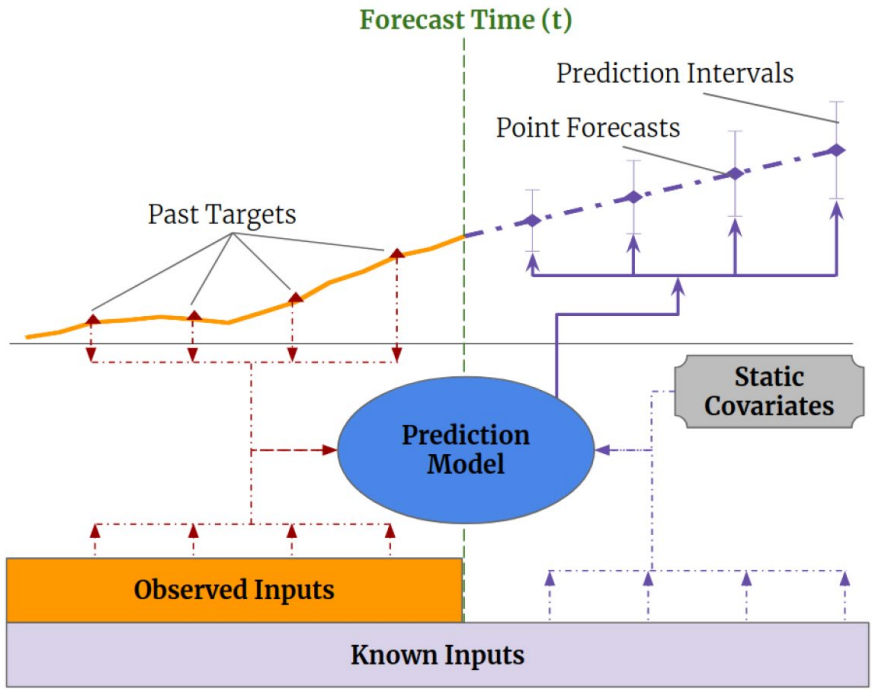


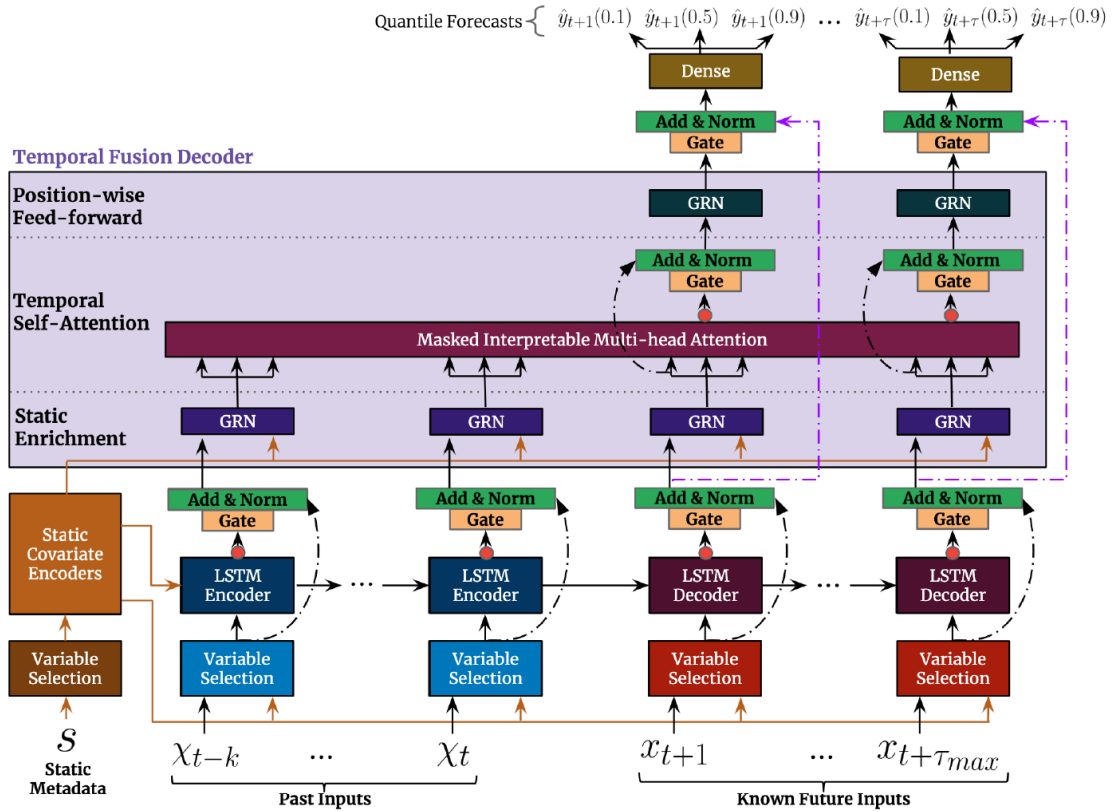


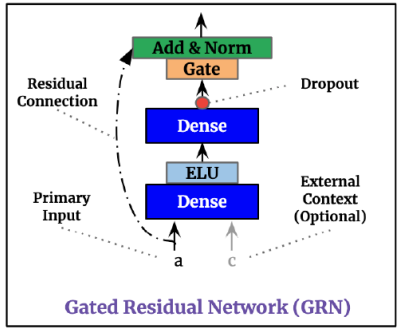
Temporal Fusion Transformers

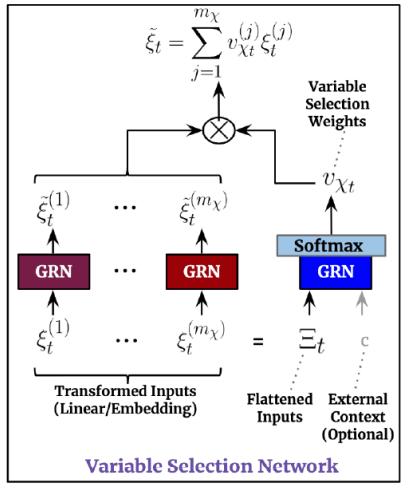
- 使用Transformer构建Encoder-Decoder结构

处理Known inputs, Observed Inputs等



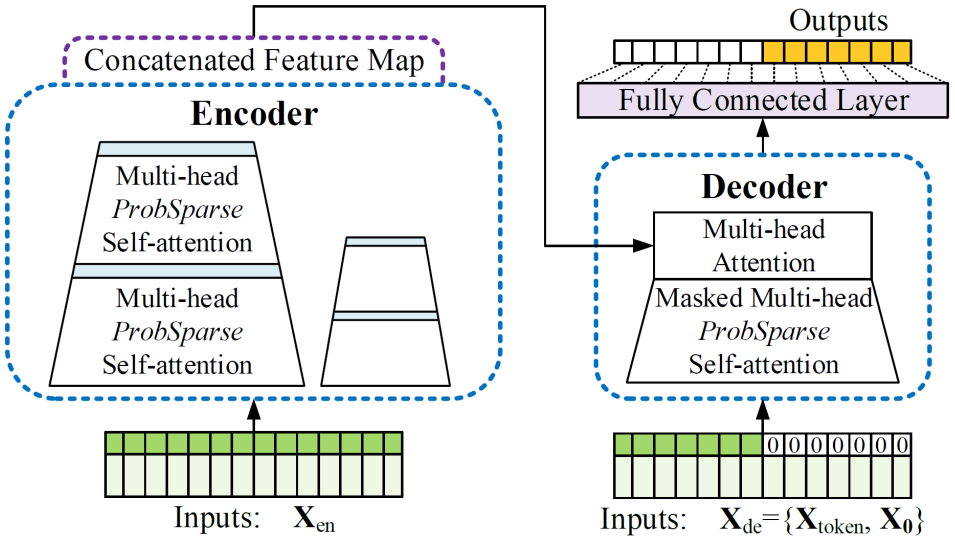




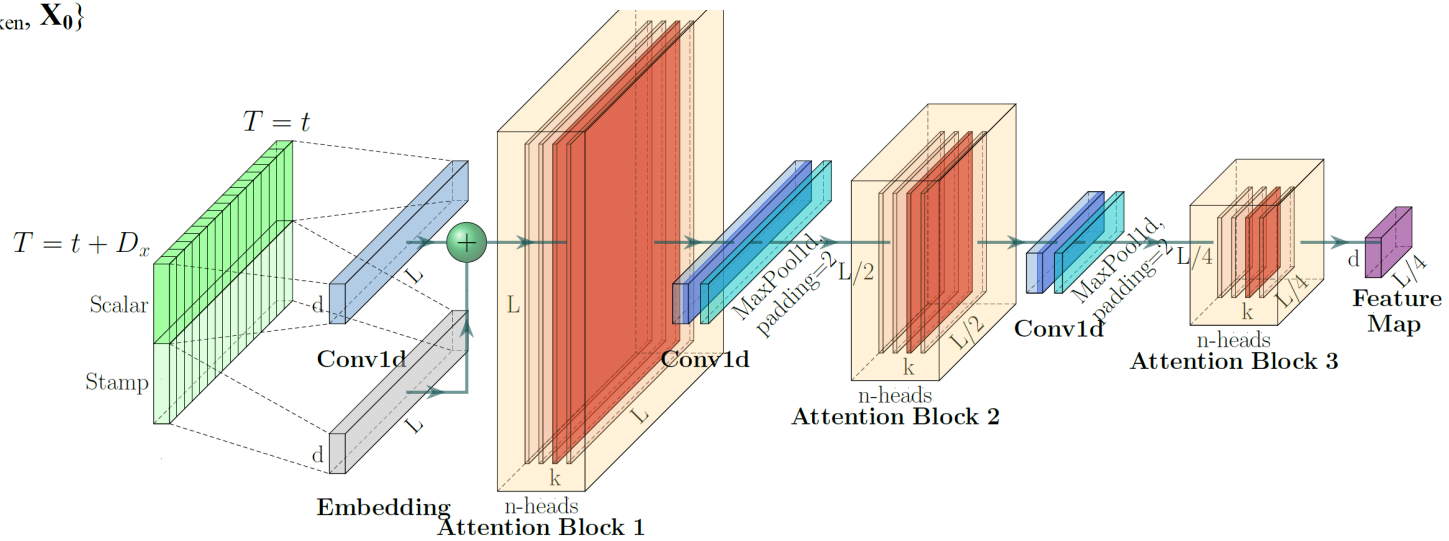


Bryan Lim et al. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. IJF 2021.

Informer



- ProbSparse self-attention for efficient and robust attention mechanism (low-rank)
- Encode timestamps as additional positional encoding by using learnable embedding layers
- Generative style decoder: produce long sequence forecasts with only one forward step, avoiding cumulative error spreading during inference



Zhou et al. **Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting**. AAAI 2021.

Transformer for Time Series

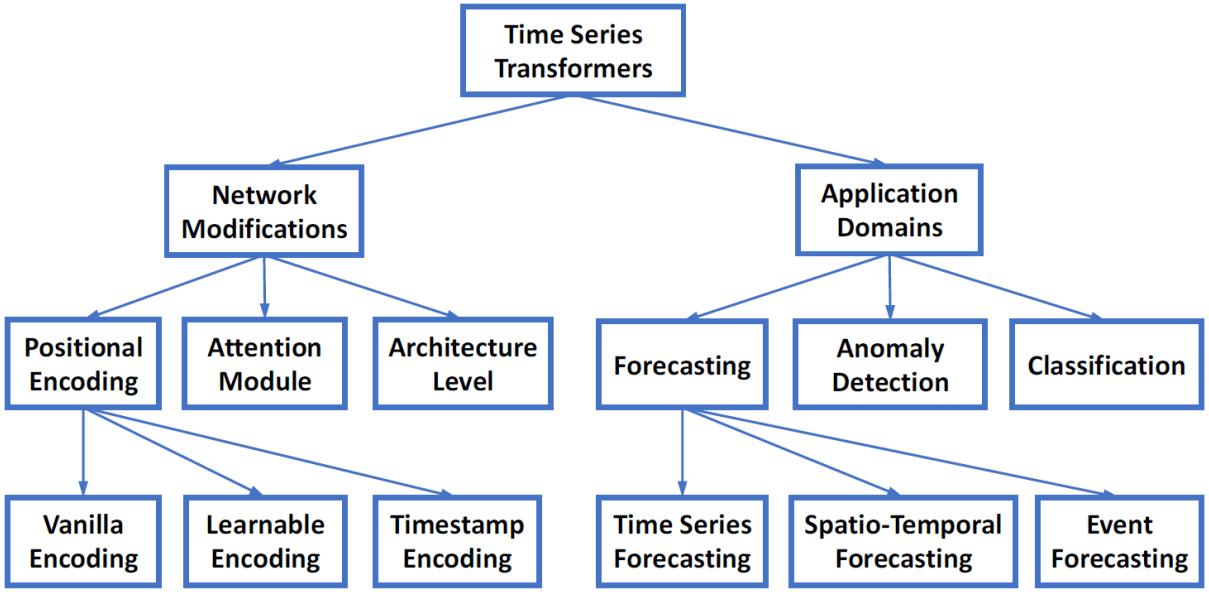


Table 1: Complexity comparisons of popular time series Transformers with different attention modules.

Methods	Training		Testing
	Time	Memory	Steps
Transformer [Vaswani *et al.*, 2017]	$\mathcal{O}\left(N^2\right)$	$\mathcal{O}\left(N^2\right)$	N
LogTrans [Li *et al.*, 2019]	$\mathcal{O}(N \log N)$	$\mathcal{O}\left(N \log N\right)$	1
Informer [Zhou *et al.*, 2021]	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Autoformer [Wu *et al.*, 2021]	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Pyraformer [Liu *et al.*, 2022a]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	1
Quatformer [Chen *et al.*, 2022]	$\mathcal{O}(2cN)$	$\mathcal{O}(2cN)$	1
FEDformer [Zhou *et al.*, 2022]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	1
Crossformer [Zhang and Yan, 2023]	$\mathcal{O}(\frac{D}{L_{seg}^2}N^2)$	$\mathcal{O}(N)$	1

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, Liang Sun. **Transformers in Time Series: A Survey**. IJCAI 2023: 6778-6786

混合模型（Hybrid Method）

- 难点
  - 1. 神经网络难以处理季节性变化，数据归一化损失信
    - Deseasonalization
    - Adaptive normalization (Exponential Smoothing, ES)
  - 2. 能够利用神经网络的强大预测能力
    - Generation of forecasts (LSTM)
    - Ensembling

hybrid approaches *and combinations of method are the way forward for improving the forecasting accuracy and making forecasting more valuable* *[Makridakis et al. 2018]*

混合模型（Hybrid Method）

非概率模型,ES-RNN

$$\hat{y}_{i,t+\tau} = exp\left(\boldsymbol{W}_{ES}\boldsymbol{h}^{L}_{i,t+\tau} + \boldsymbol{b}_{ES}\right) \times l_{i,t} \times \gamma_{i,t+\tau},$$

$$l_{i,t} = \frac{\beta_1^{(i)} y_{i,t}}{\gamma_{i,t}} + \left(1 - \beta_1^{(i)}\right) l_{i,t-1}$$

$$\gamma_{i,t} = \frac{\beta_2^{(i)} y_{i,t}}{l_{i,t}} + \left(1 - \beta_2^{(i)}\right) \gamma_{i,t-\kappa}$$

概率模型

产生概率分布参数

$$y_t = a\left(\boldsymbol{h}^{L}_{i,t+\tau}\right)^{T} \boldsymbol{l}_t + \phi\left(\boldsymbol{h}^{L}_{i,t+\tau}\right)\epsilon_t$$

$$\boldsymbol{l}_t = F\left(\boldsymbol{h}^{L}_{i,t+\tau}\right)\boldsymbol{l}_{t-1} + q\left(\boldsymbol{h}^{L}_{i,t+\tau}\right) + \Sigma\left(\boldsymbol{h}^{L}_{i,t+\tau}\right)$$

混合模型（Hybrid Method）

- 指数平滑移除季节趋势
- K为每季节周期观测样本量
  - 无季节性模型

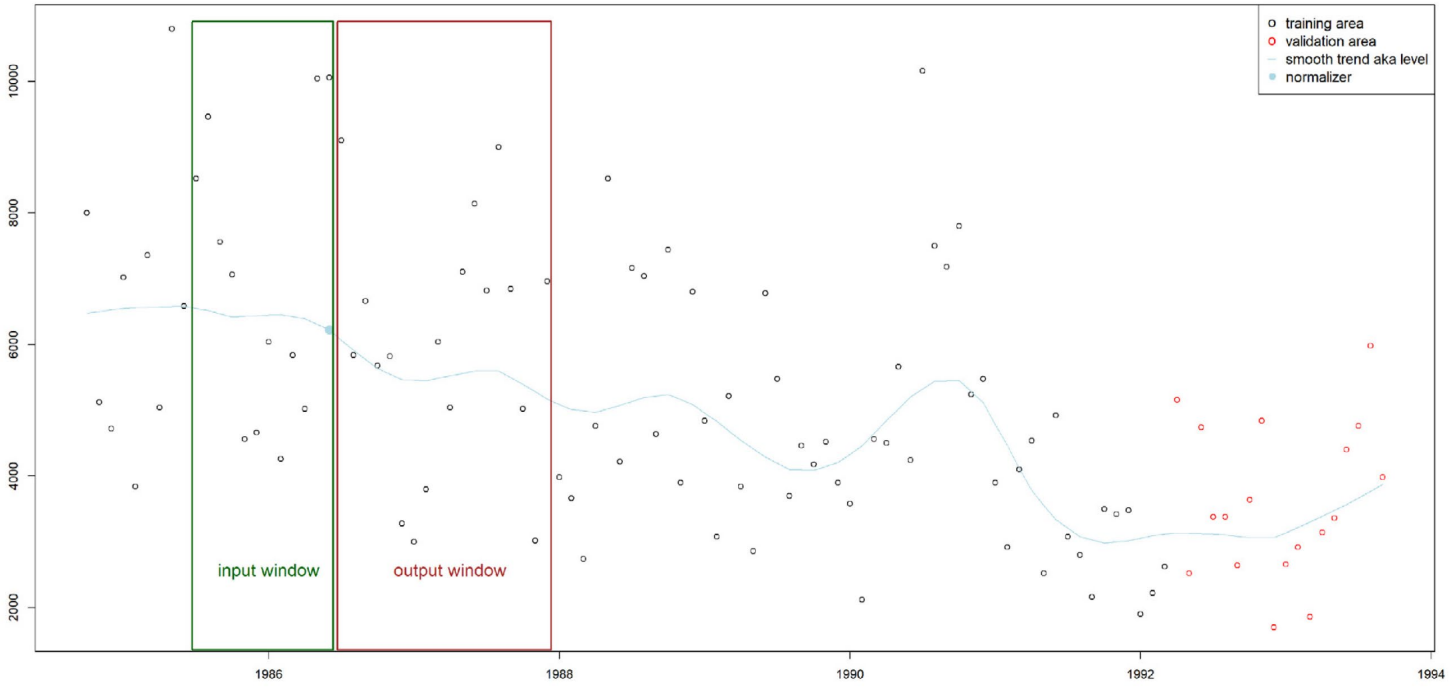
$$l_t = \alpha Y_t + (1-\alpha) l_{t-1}$$

  - 单一季节性模型（Single Seasonality）

$$l_t = \alpha Y_t / s_t + (1-\alpha) l_{t-1}$$
$$s_{t+K} = \beta Y_t / l_t + (1-\beta) s_t$$

  - 双重季节性模型（Double Seasonality）

$$l_t = \alpha Y_t / (s_t u_t) + (1-\alpha) l_{t-1}$$
$$s_{t+K} = \beta Y_t / (l_t u_t) + (1-\beta) s_t$$
$$u_{t+L} = \gamma Y_t / (l_t s_t) + (1-\gamma) u_t,$$



- 1. 使用输入窗口（input window）最后一个数值做归一化
- 2. 除以季节性成分
- 3. 使用Log变换

Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. ***International Journal of Forecasting***. 2020.

混合模型（Hybrid Method）

- 神经网络预测（ LSTM ）
- 非季节性

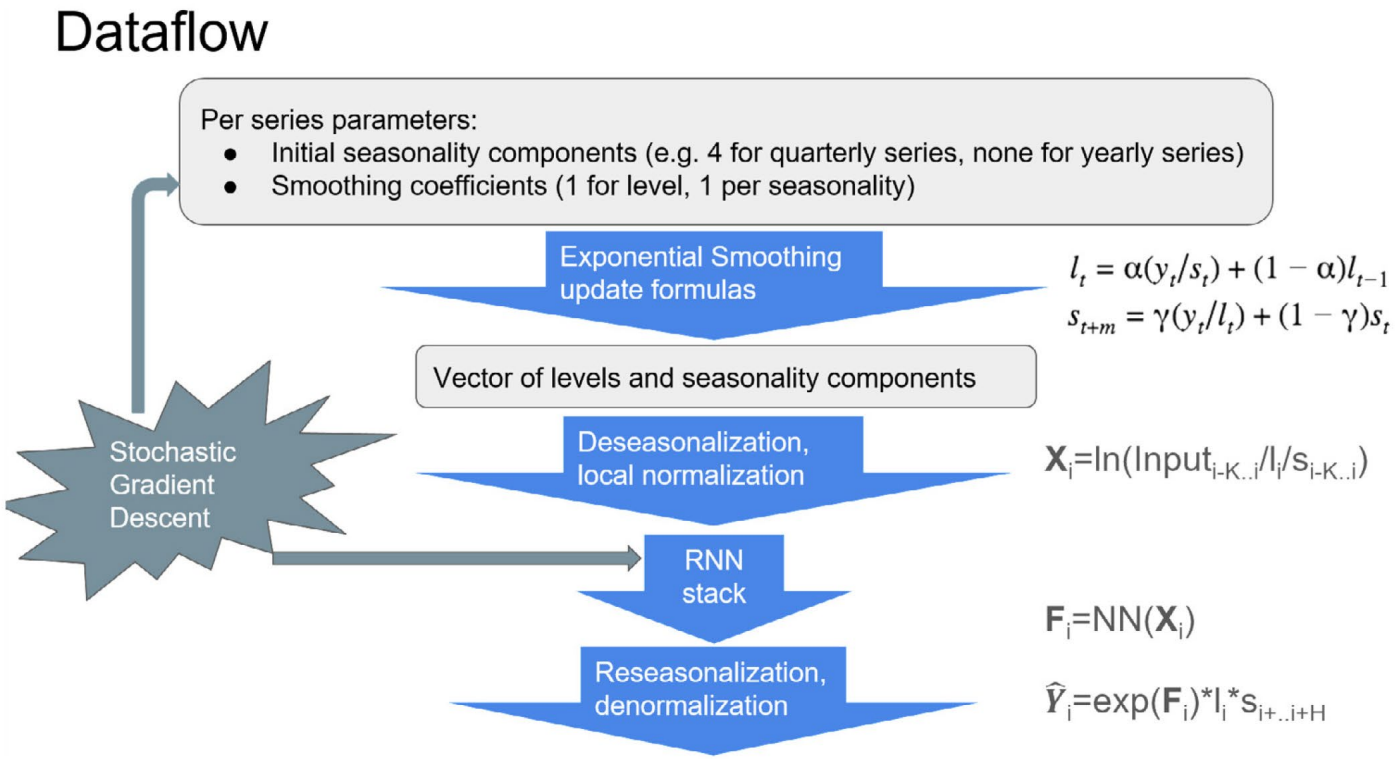
$$\hat{y}_{t+1..t+h} = \exp(NN(x)) * l_t$$

- 单一季节性

$$\hat{y}_{t+1..t+h} = \exp(NN(x)) * l_t * s_{t+1...t+h}$$

- 双重季节性模型（Double Seasonality）

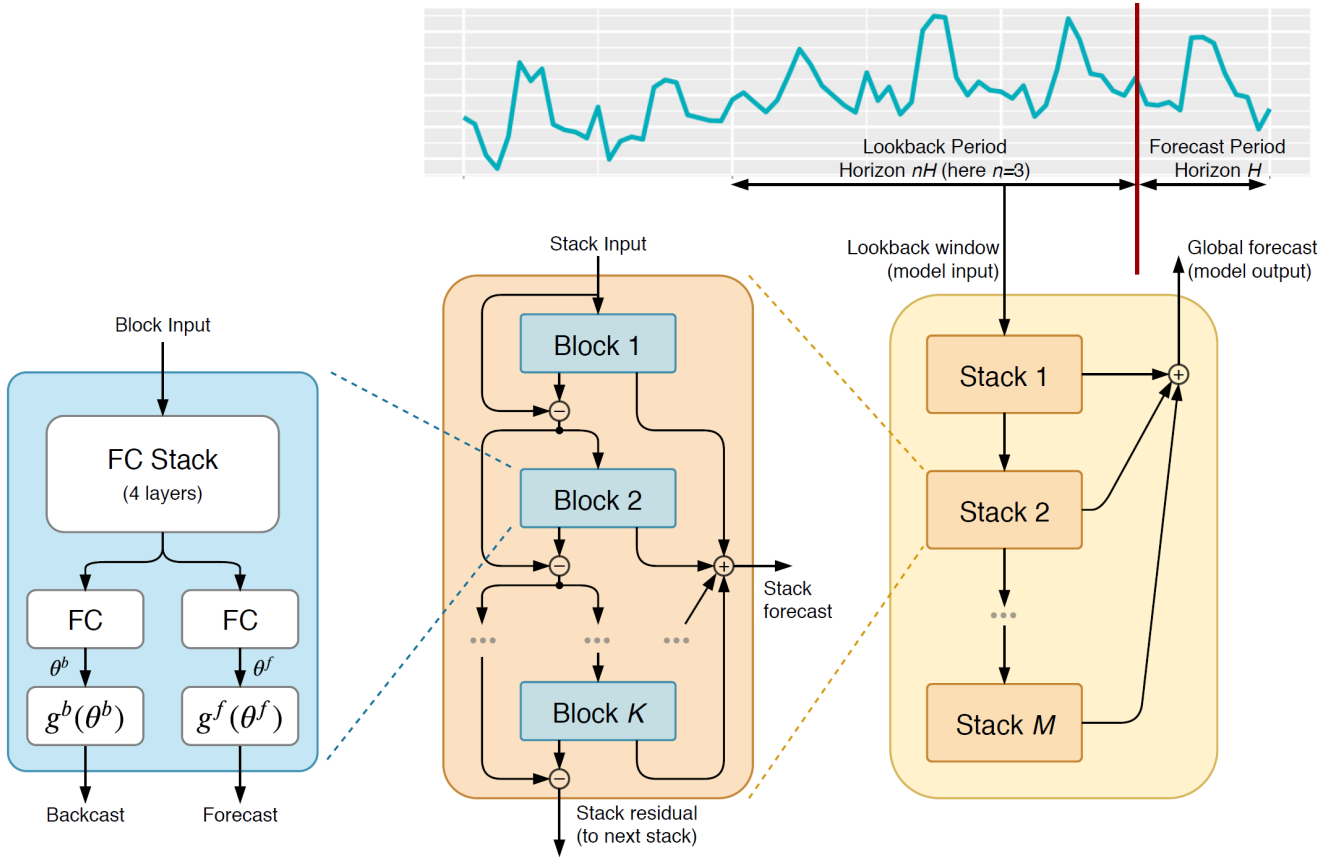
$$\hat{y}_{t+1..t+h} = \exp(NN(x)) * l_t * s_{t+1:..t+h} * u_{t+1:..t+h},$$



N-BEATS

- 传统时间序列方法 vs. 机器学习方法
- 直接学习序列的各个要素
- 什么样的模型结构能够便于时序预测

*Our **model does not employ any time-series-specific components** and its performance on heterogeneous datasets strongly suggests that, contrarily to received wisdom, deep learning primitives such as residual blocks are **by themselves sufficient to solve a wide range of forecasting problems***

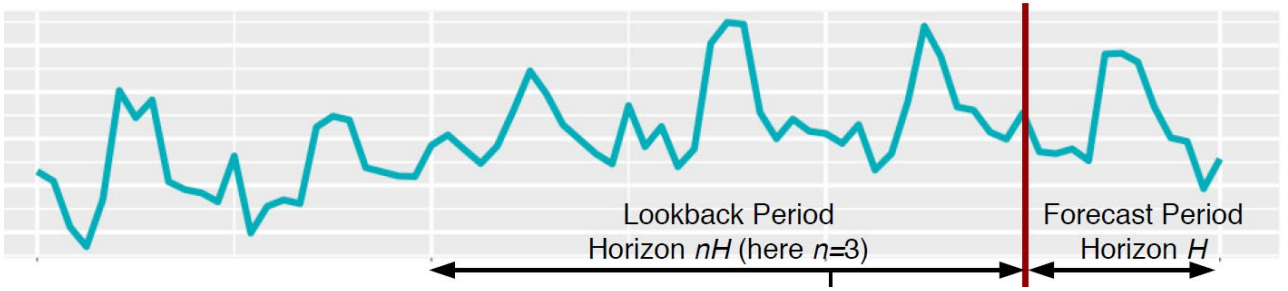


Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, Yoshua Bengio. **N-BEATS: Neural basis expansion analysis for interpretable time series forecasting**. *ICLR* 2020

N-BEATS - 基本单元

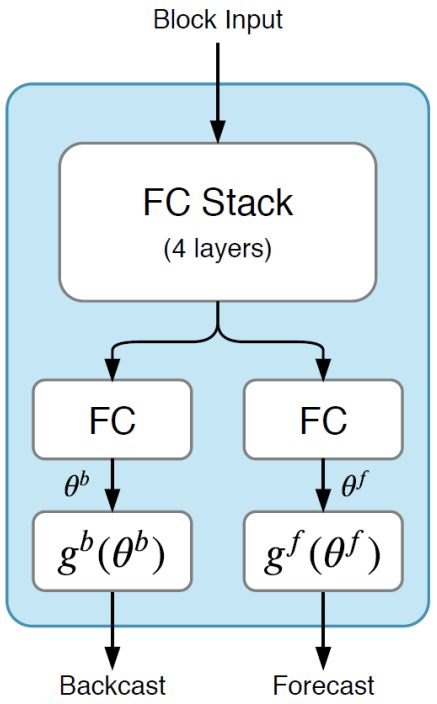
- 输入为$\mathbf{x}_\ell$为一定范围内（nH）的历史数据，产生加权系数

$$\mathbf{h}_{\ell,1} = \mathrm{FC}_{\ell,1}(\mathbf{x}_\ell), \mathbf{h}_{\ell,2} = \mathrm{FC}_{\ell,2}(\mathbf{h}_{\ell,1}),$$
$$\mathbf{h}_{\ell,3} = \mathrm{FC}_{\ell,3}(\mathbf{h}_{\ell,2}), \mathbf{h}_{\ell,4} = \mathrm{FC}_{\ell,4}(\mathbf{h}_{\ell,3})$$
$$\theta_\ell^b = \mathrm{LINEAR}_\ell^b(\mathbf{h}_{\ell,4}),$$
$$\theta_\ell^f = \mathrm{LiNEAR}_\ell^f(\mathbf{h}_{\ell,4})$$

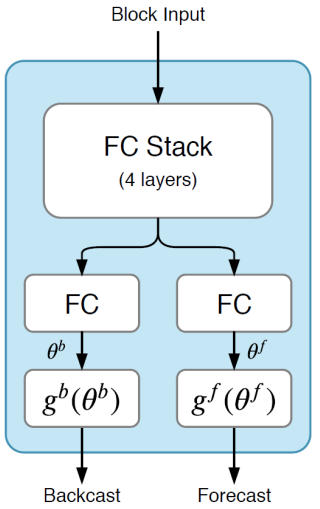


- 输出：使用针对基（Basis Layers）的加权组合
  - 1. g^f：长度为H的预测$\hat{\mathbf{y}}_\ell$，forecast
  - 2. g^b：对$\mathbf{x}_\ell$的最优估计$\hat{\mathbf{x}}_\ell$，backcast

$$\hat{\mathbf{y}}_\ell = \sum_{i=1}^{\dim(\theta_\ell^f)} \theta_{\ell,i}^f \mathbf{v}_i^f, \qquad \hat{\mathbf{x}}_\ell = \sum_{i=1}^{\dim(\theta_\ell^b)} \theta_{\ell,i}^b \mathbf{v}_i^b$$



N-BEATS – 基的设定



- 基于基输出（g^b和g^f的实现）

$$\hat{\mathbf{y}}_\ell = \sum_{i=1}^{\dim\left(\theta_\ell^f\right)} \theta_{\ell,i}^f \mathbf{v}_i^f, \qquad \hat{\mathbf{x}}_\ell = \sum_{i=1}^{\dim\left(\theta_\ell^b\right)} \theta_{\ell,i}^b \mathbf{v}_i^b$$

- 趋势模型，使用多项式建模

$$\hat{\mathbf{y}}_{s,\ell} = \sum_{i=0}^{p} \theta_{s,\ell,i}^f t^i$$

- 定义，先前预测H步，

$$\hat{\mathbf{y}}_{s,\ell}^{tr} = \mathbf{T}\theta_{s,\ell}^f$$

其中$\mathbf{t} = [0,1,2,\dots,H-2,H-1]^T/H$，
$\mathbf{T} = [\mathbf{1},\mathbf{t},\dots,\mathbf{t}^p]$

- 季节模型，基于傅里叶级数

$$\hat{\mathbf{y}}_{s,\ell} = \sum_{i=0}^{\lfloor H/2-1 \rfloor} \theta_{s,\ell,i}^f \cos(2\pi i t) + \theta_{s,\ell,i+\lfloor H/2 \rfloor}^f \sin(2\pi i t)$$

- 定义，先前预测H步，

$$\hat{\mathbf{y}}_{s,\ell}^{seas} = \mathbf{S}\theta_{s,\ell}^f$$

$\mathbf{S}$
$= [\mathbf{1}, \cos(2\pi\mathbf{t}), \dots \cos(2\pi\lfloor H/2-1 \rfloor \mathbf{t})), \sin(2\pi\mathbf{t}), \dots, \sin(2\pi\lfloor H/2-1 \rfloor \mathbf{t}))]$

N-BEATS – 单元连接

- （逐层）双重残差结构
  - backcast

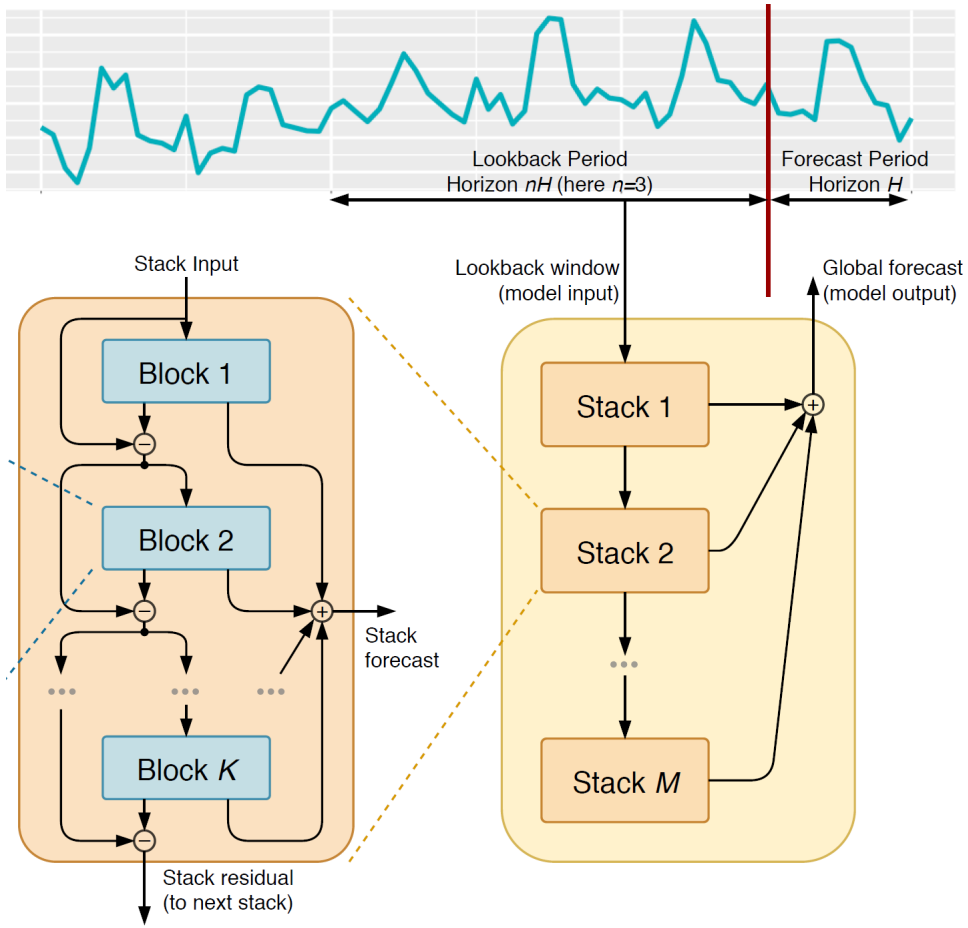
$$\mathbf{x}_\ell = \mathbf{x}_{\ell-1} - \hat{\mathbf{x}}_{\ell-1},$$

从原始信号中**移除**$\hat{\mathbf{x}}_{\ell-1}$

  - forecast

$$\hat{\mathbf{y}} = \sum_\ell \hat{\mathbf{y}}_\ell$$

逐层预测的累积



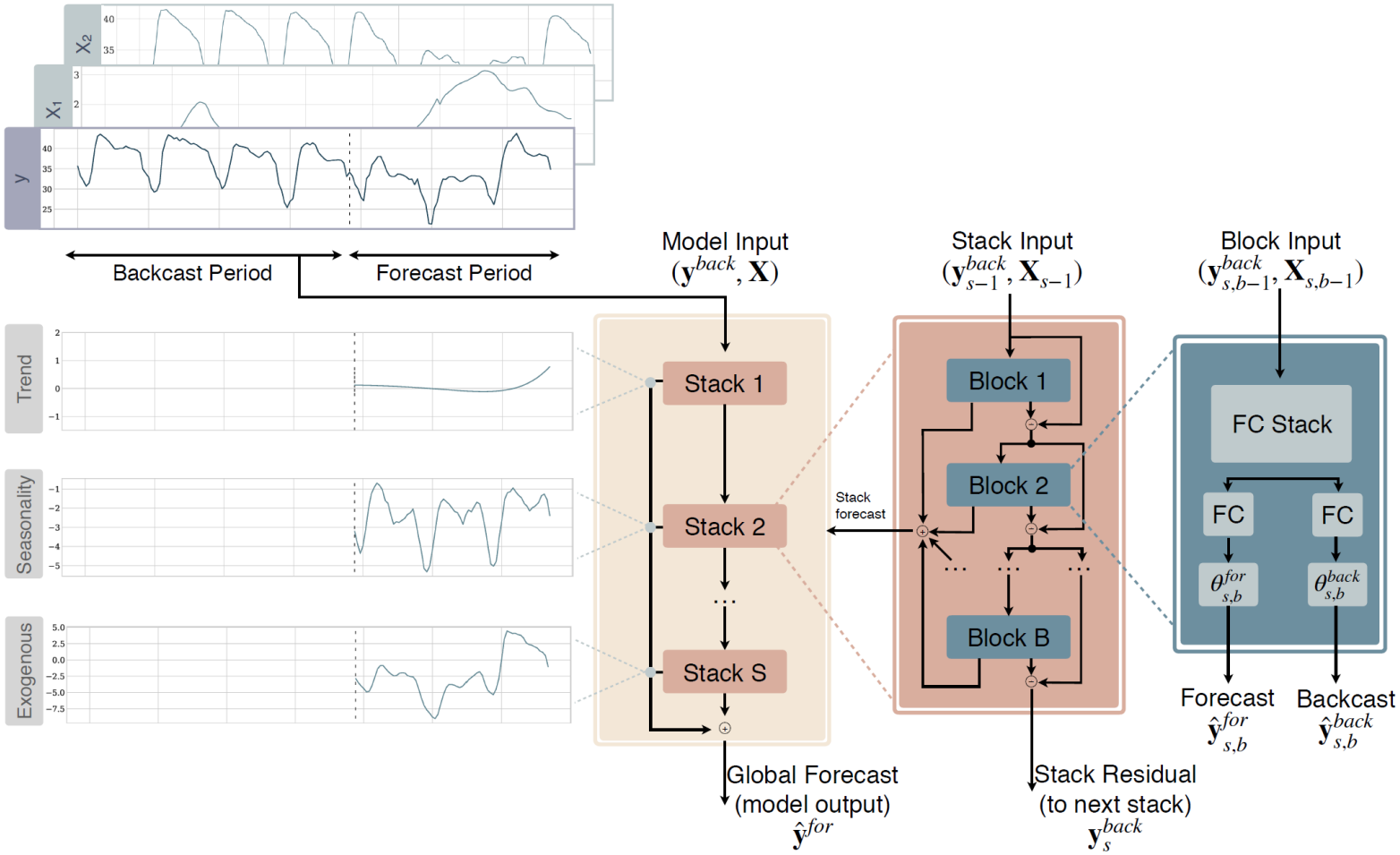
N-BEATS评估

- generic (N-BEATS-G)
- interpretable (N-BEATS-I),
- N-BEATS-I+G (ensemble of all models from N-BEATS-G and N-BEATS-I)

M4 Average (100,000)			M3 Average (3,003)	
	sMAPE	OWA		sMAPE
Pure ML	12.894	0.915	Comb S-H-D	13.52
Statistical	11.986	0.861	ForecastPro	13.19
ProLogistica	11.845	0.841	Theta	13.01
ML/TS combination	11.720	0.838	DOTM	12.90
DL/TS hybrid	11.374	0.821	EXP	12.71
N-BEATS-G	11.168	0.797		12.47
N-BEATS-I	11.174	0.798		12.43
N-BEATS-I+G	**11.135**	**0.795**		**12.37**

N-BEATS扩展

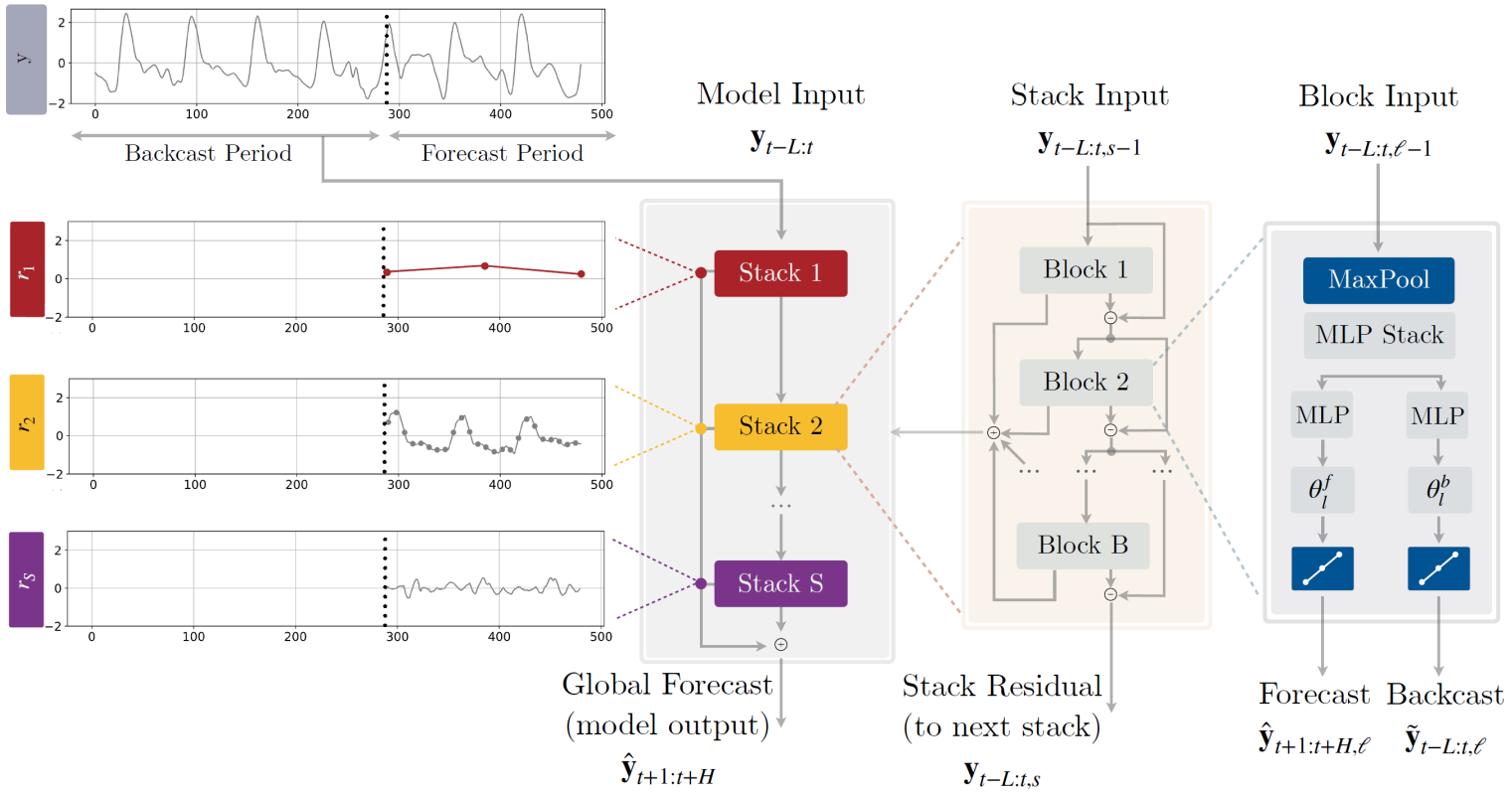
- Extend N-BEATS to incorporate exogenous factors



Kin G. Olivares et al. **Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx**. *IJF*. 2020.

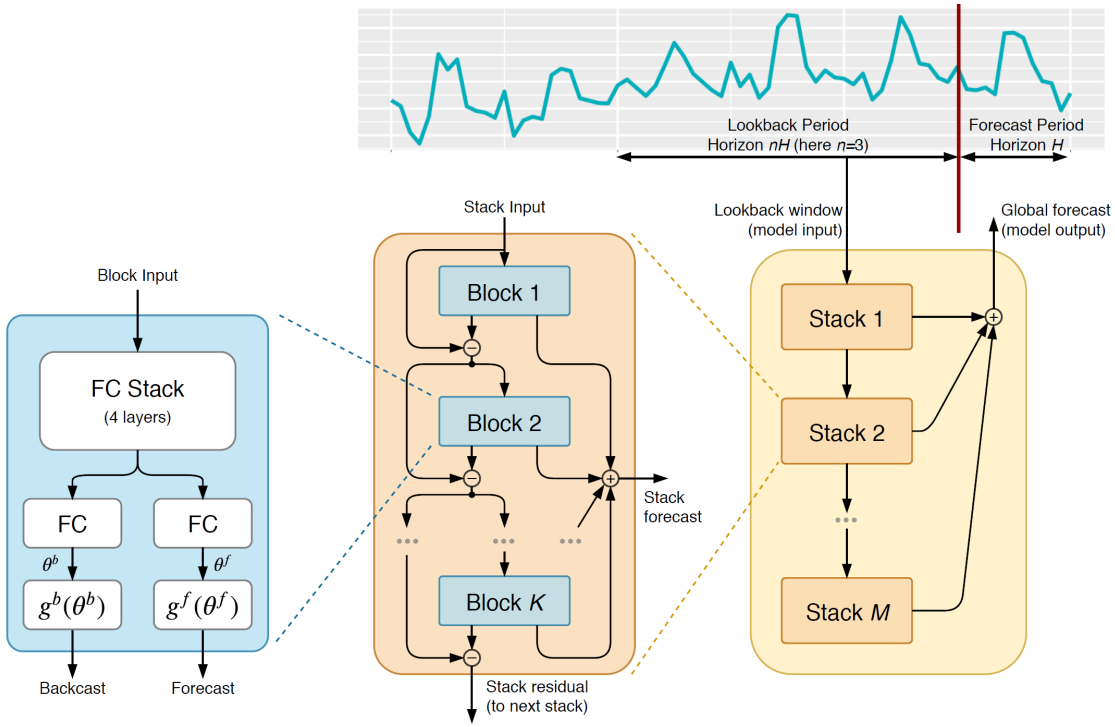
N-BEATS扩展

- NHITS: use multi-rate sampling of the input signal and multi-scale synthesis of the forecast, resulting in a hierarchical construction of forecast, greatly reducing computational requirements and improving forecasting accuracy



Cristian Challu et al. **NHITS: Neural Hierarchical Interpolation for Time Series Forecasting**. AAAI 2023: 6989-6997

N-BEATS的元学习视角



$$f: \mathbf{x}_\ell \mapsto \mathbf{h}_{\ell,4}; \quad g: \mathbf{h}_{\ell,4} \mapsto \hat{\mathbf{y}}_\ell; \quad q: \mathbf{h}_{\ell,4} \mapsto \hat{\mathbf{x}}_\ell$$

N-BEATS表示为

Inner Loop

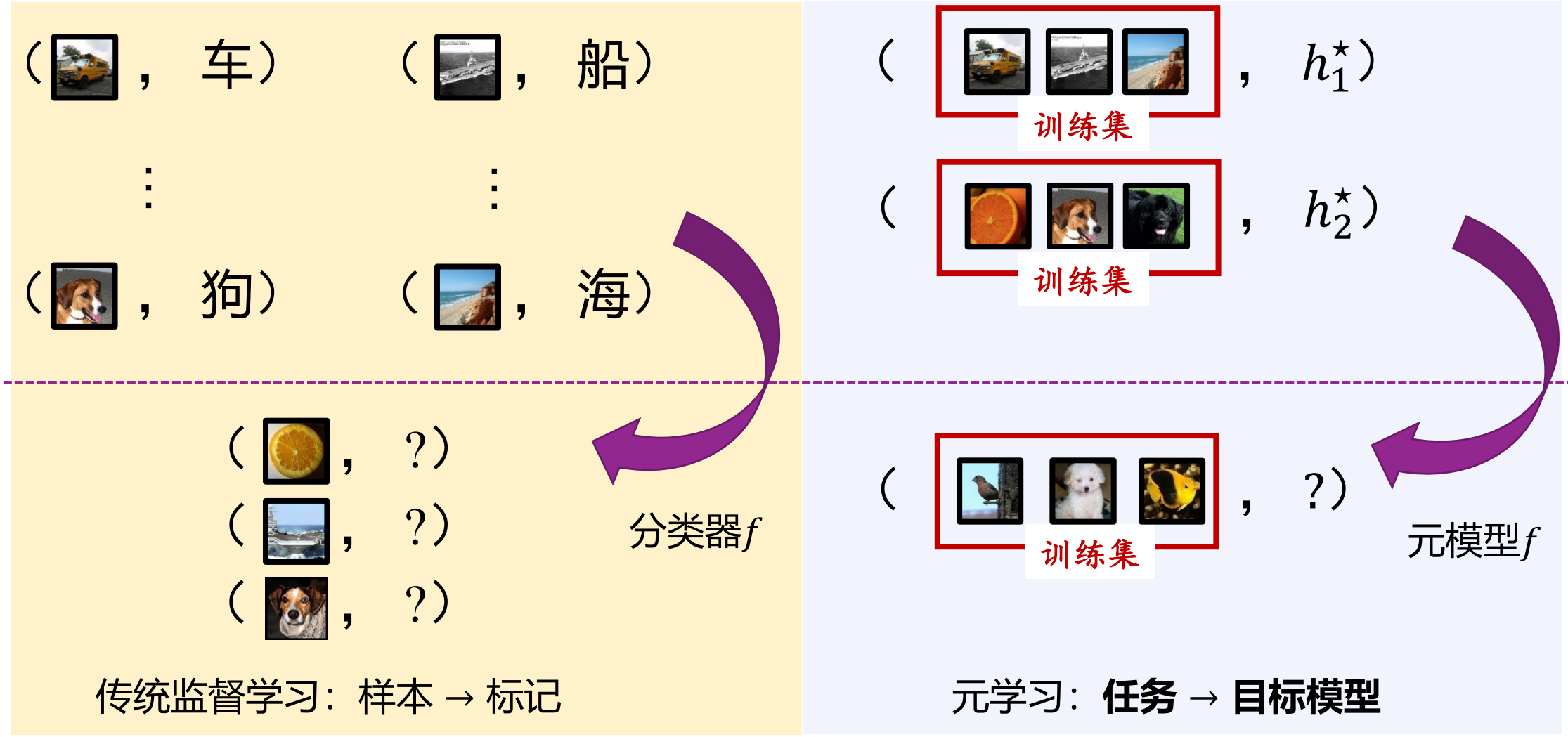
$$\hat{\mathbf{y}} = g \circ f(\mathbf{x}) + \sum_{\ell>1} g \circ f\left(\mathbf{x}_{\ell-1} - q \circ f(\mathbf{x}_{\ell-1})\right)$$

Zero-Shot Forecasting

M4, sMAPE		M3, sMAPE‡		TOURISM, MAPE	
Pure ML	12.89	Comb	13.52	ETS	20.88
Best STAT	11.99	ForePro	13.19	Theta	20.88
ProLogistica	11.85	Theta	13.01	ForePro	19.84
Best ML/TS	11.72	DOTM	12.90	Strato	19.52
DL/TS hybrid	11.37	EXP	12.71	LCBaker	19.35
N-BEATS	11.14		12.37		18.52
DeepAR*	12.25		12.67		19.27
DeepAR-M4*	n/a		14.76		24.79
N-BEATS-M4	n/a		12.44		18.82
N-BEATS-FR	11.70		12.69		19.94

Boris N. Oreshkin et al. **Meta-learning framework with applications to zero-shot time-series forecasting**. AAAI 2021

Meta-Learning



- 训练模式
- 模型集成
- 任务迁移

模型集成

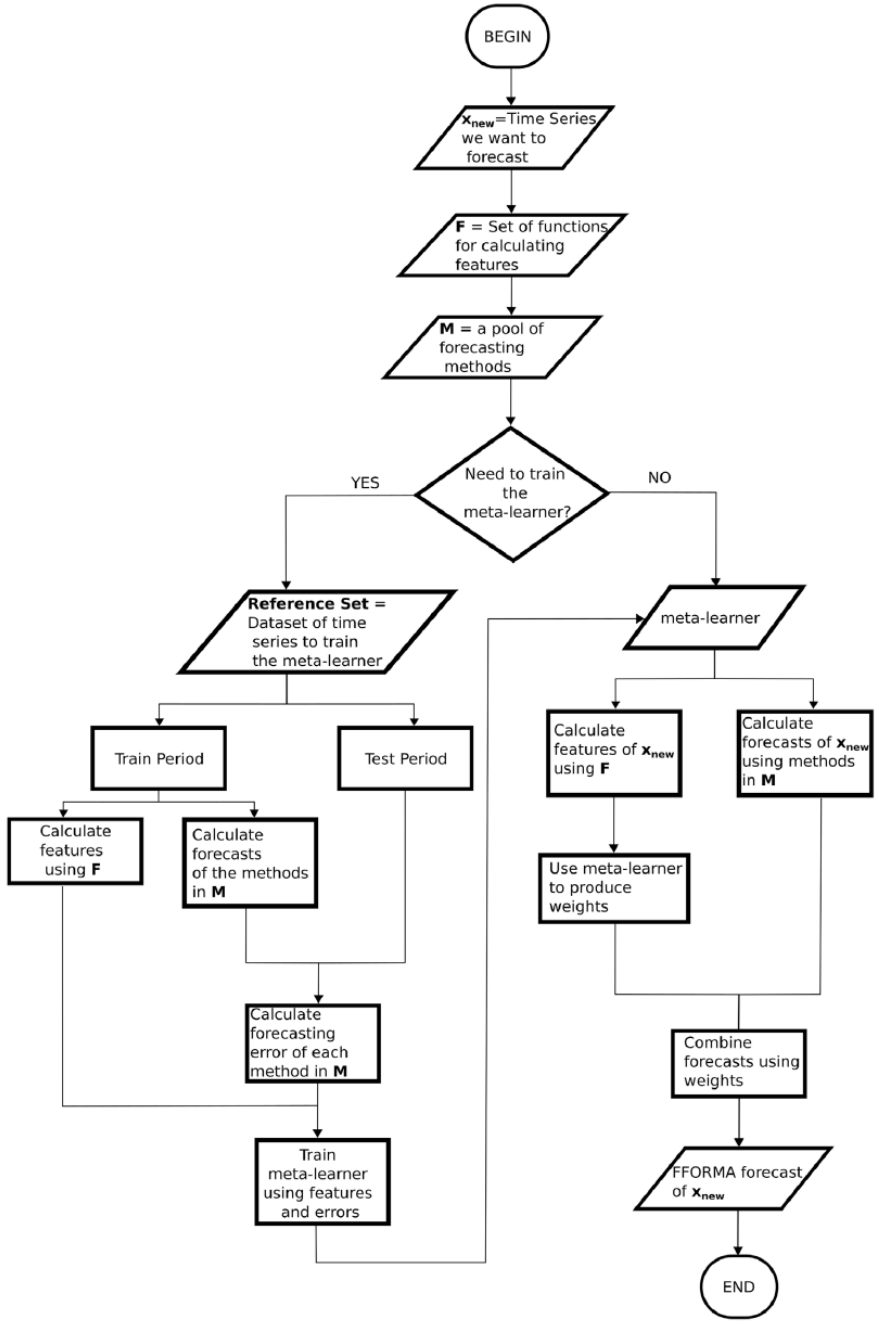
- 如何确定不同预测模型的权重？Meta-Learn the weights

Model Zoo

- Naïve
- random walk with drift
- seasonal naïve
- theta method
- automated ARIMA
- automated exponential smoothing
- TBATS model
- STLM-AR
- neural network time series forecasts

- 难点：

  基于什么确定加权？



Pablo Montero-Manso et al. FFORMA: Feature-based forecast model averaging. IJF 2020. (Rank 2nd in M4)

Algorithm 1 The FFORMA framework: Forecast combination based on meta-learning

OFFLINE PHASE: TRAIN THE LEARNING MODEL

Inputs:

$\{x_1, x_2, \ldots, x_N\}$: N observed time series that form the reference set.
F: a set of functions for calculating time series features.
M: a set of forecasting methods in the pool, e.g., naïve, ETS, ARIMA, etc.

Output:

FFORMA meta-learner: A function from the extracted features to a set of M weights, one for each forecasting method.

Prepare the meta-data

1: **for** $n = 1$ to N: **do**
2: Split x_n into a training period and test period.
3: Calculate the set of features $\boldsymbol{f}_n \in F$ over the training period.
4: Fit each forecasting method $m \in M$ over the training period and generate forecasts over the test period.
5: Calculate forecast losses L_{nm} over the test period.
6: **end for**

Train the meta-learner, w

7: Train a learning model based on the meta-data and errors, by minimizing:

$$\underset{w}{\operatorname{argmin}} \sum_{n=1}^{N} \sum_{m=1}^{M} w(\boldsymbol{f}_n)_m L_{nm}.$$

基于XGBoost

ONLINE PHASE: FORECAST A NEW TIME SERIES

Input:

FFORMA meta-learner from offline phase.

Output:

Forecast the new time series x_{new}.

8: **for** each x_{new}: **do**
9: Calculate features $\boldsymbol{f}_{new}$ by applying F.
10: Use the meta-learner to produce $\boldsymbol{w}(\boldsymbol{f}_{new})$, an M-vector of weights.
11: Compute the individual forecasts of the M forecasting methods in the pool.
12: Combine the individual forecasts using $\boldsymbol{w}$ to generate final forecasts.
13: **end for**

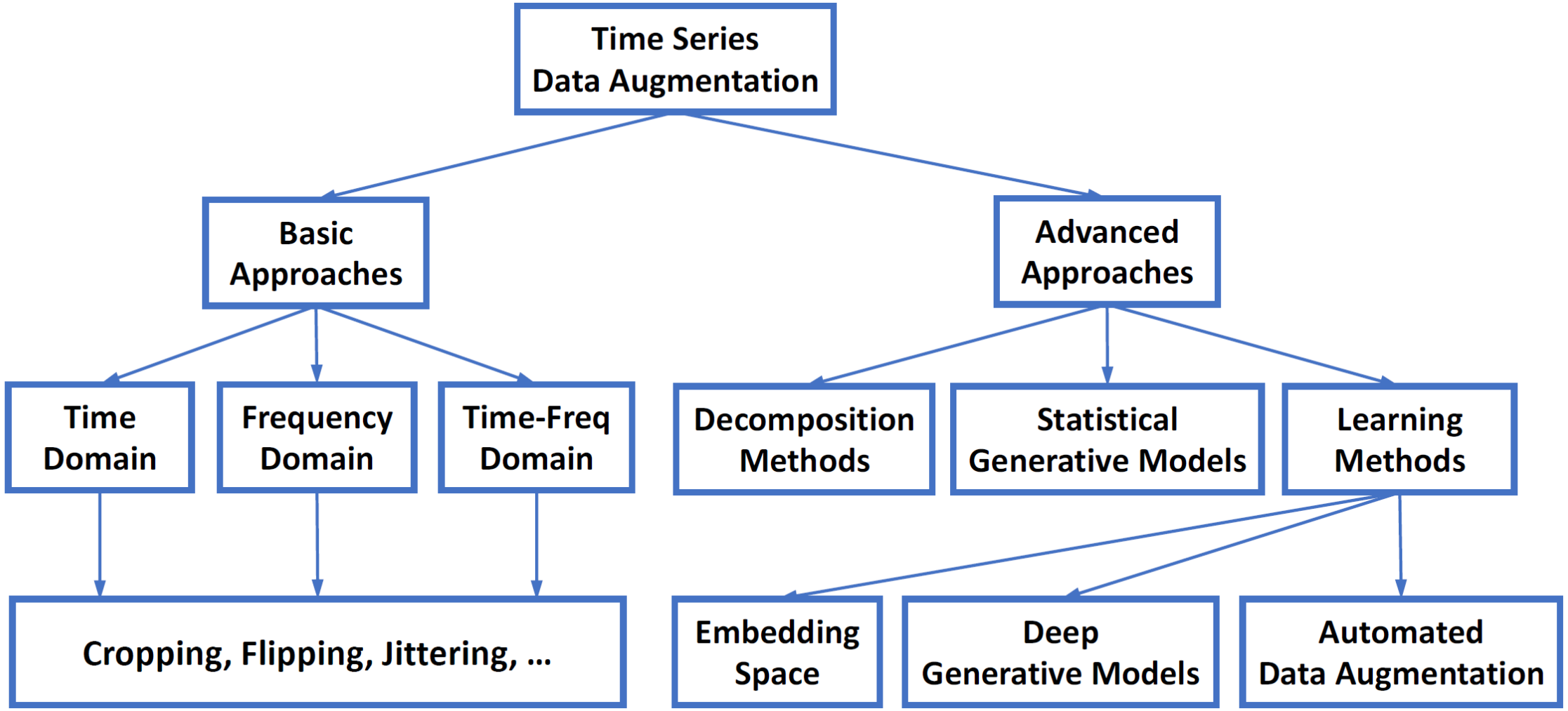
构造Meta-Feature

Features used in the FFORMA framework.

Feature		Description	Non-seasonal	Seasonal
1	T	length of time series	✓	✓
2	trend	strength of trend	✓	✓
3	seasonality	strength of seasonality	–	✓
4	linearity	linearity	✓	✓
5	curvature	curvature	✓	✓
6	spikiness	spikiness	✓	✓
7	e_acf1	first ACF value of remainder series	✓	✓
8	e_acf10	sum of squares of first 10 ACF values of remainder series	✓	✓
9	stability	stability	✓	✓
10	lumpiness	lumpiness	✓	✓
11	entropy	spectral entropy	✓	✓
12	hurst	Hurst exponent	✓	✓
13	nonlinearity	nonlinearity	✓	✓
13	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓
14	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓
15	hwalpha	ETS(A,A,A) $\hat{\alpha}$	–	✓
16	hwbeta	ETS(A,A,A) $\hat{\beta}$	–	✓
17	hwgamma	ETS(A,A,A) $\hat{\gamma}$	–	✓
18	ur_pp	test statistic based on Phillips–Perron test	✓	✓
19	ur_kpss	test statistic based on KPSS test	✓	✓
20	y_acf1	first ACF value of the original series	✓	✓
21	diff1y_acf1	first ACF value of the differenced series	✓	✓
22	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓
23	y_acf10	sum of squares of first 10 ACF values of original series	✓	✓
24	diff1y_acf10	sum of squares of first 10 ACF values of differenced series	✓	✓
25	diff2y_acf10	sum of squares of first 10 ACF values of twice-differenced series	✓	✓
26	seas_acf1	autocorrelation coefficient at first seasonal lag	–	✓
27	sediff_acf1	first ACF value of seasonally differenced series	–	✓
28	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓
29	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓
30	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓
31	seas_pacf	partial autocorrelation coefficient at first seasonal lag	✓	✓
32	crossing_point	number of times the time series crosses the median	✓	✓
33	flat_spots	number of flat spots, calculated by discretizing the series into 10 equal-sized intervals and counting the maximum run length within any single interval	✓	✓
34	nperiods	number of seasonal periods in the series	–	✓
35	seasonal_period	length of seasonal period	–	✓
36	peak	strength of peak	✓	✓
37	trough	strength of trough	✓	✓
38	ARCH.LM	ARCH LM statistic	✓	✓
39	arch_acf	sum of squares of the first 12 autocorrelations of z^2	✓	✓
40	garch_acf	sum of squares of the first 12 autocorrelations of r^2	✓	✓
41	arch_r2	R^2 value of an AR model applied to z^2	✓	✓
42	garch_r2	R^2 value of an AR model applied to r^2	✓	✓

Christiane Lemke, Bogdan Gabrys. Meta-learning for time series forecasting and forecast combination. Neurocomputing 2010. (Based on Decision Tree)

Data Augmentation



Qingsong Wen et al. **Time Series Data Augmentation for Deep Learning: A Survey**. IJCAI 2021: 4653-4660