# Time Series Introduction

**Michalis Vazirgiannis**

**Data Science & Mining group**
LIX, Ecole Polytechnique,
DASCIM web page: http://www.lix.polytechnique.fr/dascim
Google Scholar: https://bit.ly/2rwmvQU
Twitter: @mvazirg

**June 2021**

# Outline

**Introduction to Time Series**

**Necessary mathematical background**

- time series as a stochastic process
- White Noise and Random Walk
- Autocovariance
- Autocorrelation
- Stationarity
- Autocorrelation for Stationary Time Series
- Partial Autocorrelation

**Time-domain models**

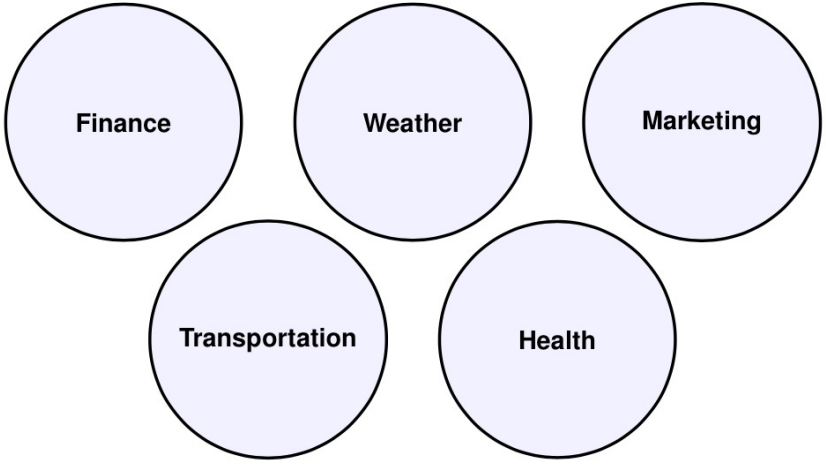
- Autoregressive Models (AR)
- Moving Average Models (MA)
- Autoregressive - Moving Average Models (ARMA)
- ARIMA Models

**Deep Learning approaches**

# Motivation

## *Why focus on Times Series models?*

1. **Constant need to predict the future.**
2. **Focus on minimizing risk, developing long-term strategies, making decisions instantly.**
3. **Availability of a great amount of data.**
4. **Impact on a huge range of applications.**



# Application Domains - Examples

Forecasting the

- new cases of covid-19 in a city each day.
- closing price of a stock each minute.
- product sales in units sold each day for a store.
- birth rate at all hospitals in a city each year.
- utilization demand on a server each hour.
- average price of gasoline in a city each day.
- number of passengers through a train station each day.

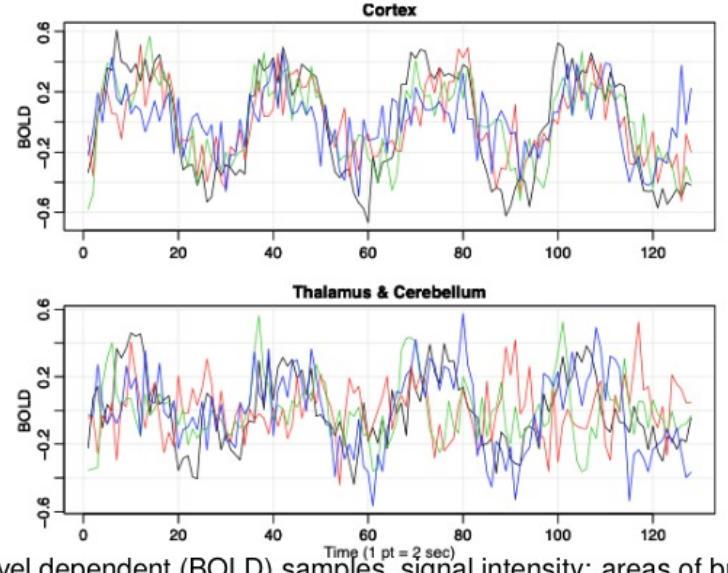
# A Time Series Example - Stock prices

APPLE, TESLA stocks, A 5-year weekly evolution[1]


[1] *source: finance.yahoo.com*

# A Time Series Example - fMRI Imaging

- fMRI data from various locations in the cortex, thalamus, and cerebellum; n = 128points, one observation taken every 2 seconds

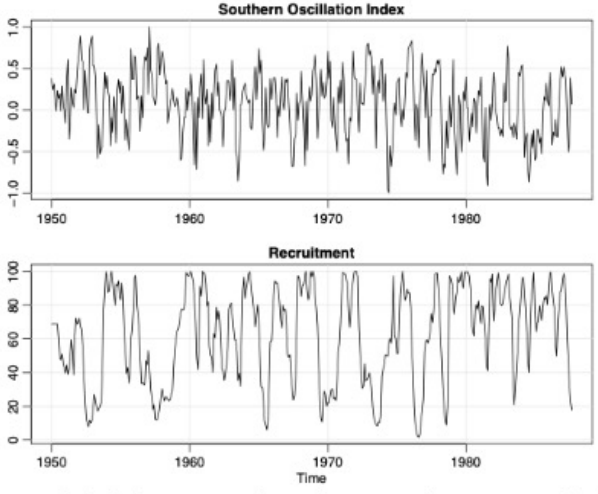



- blood oxygenation-level dependent (BOLD) samples, signal intensity: areas of brain activation in the brain.
- periodicities appear strongly in motor cortex series, less strongly in the thalamus and cerebellum.
- series from different areas of the brain: testing whether the areas responding differently stimulus.
- Analysis of variance techniques: spectral analysis of variance.

[2] source: R. H. Shumway, D. S. Stoffer,, Time Series Analysis and Its Applications, With R Examples. Springer Texts in Statistics

# A Time Series Example - weather cycles and fish

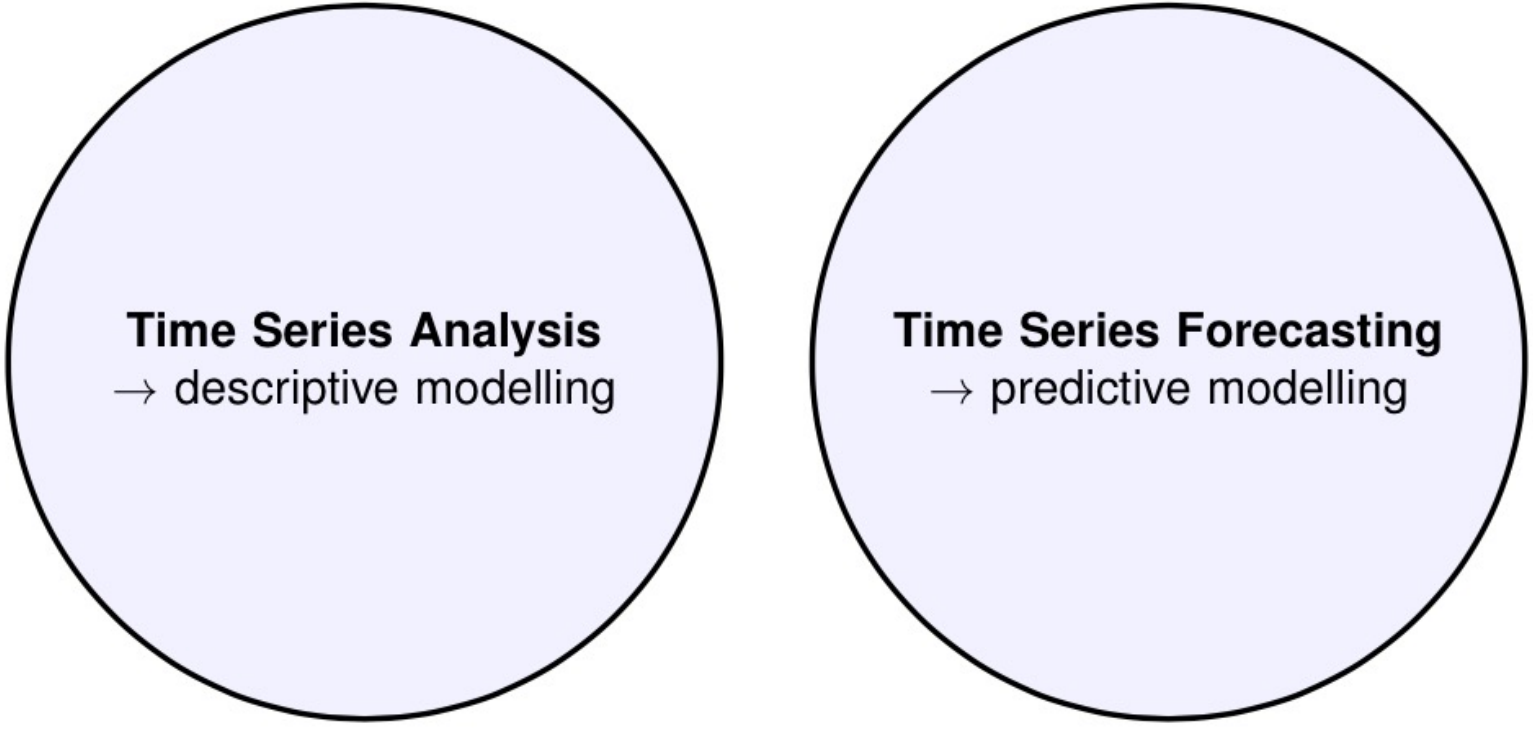
Monthly SOI and Recruitment (estimated new fish), 1950-1987 [4]




- SOI: changes in air pressure, related to sea surface temperatures - central Pacific Ocean.
- Central Pacific warms every 3-7 years (El Niño effect blamed for global extreme weather).
- Both series exhibit repetitive behavior, (cycles) - underlying processes: rate or frequency of oscillation

[4] R. H. Shumway, D. S. Stoffer, Time Series Analysis and Its Applications, With R Examples. Springer Texts in Statistics

# Time Series - Domains of study



# Time Series Analysis

- developing *models* that best capture or describe an observed time series in order **to understand the underlying causes**. Seek the **"why"***model* behind a time series dataset.
- Necessary assumptions about the form of the data and decomposing the time series into constituting components.
- Quality of a descriptive model: i. how well it describes all available data ii. the quality of the interpretation it provides for the problem domain.

***"The primary objective of time series analysis is to develop mathematical models that provide plausible descriptions from sample data."*** [5]

[5] Time Series Analysis: With Applications in R, (Springer Texts in Statistics) 2nd Edition, Cryer, Chan

# Time Series Forecasting

- Making *predictions* about the future: called extrapolation in the classical statistical handling of time series data.
- More modern fields focus on the topic and refer to it as time series forecasting.
- Forecasting: *models to fit on historical data and using them to predict future observations*.
- Descriptive models cannot predict future (i.e. to smooth or remove noise), they only seek to best describe the data.
- In forecasting future is not known and must only be estimated from the past samples.

***"The purpose of time series forecasting is generally two-fold:***
***1. model the stochastic mechanisms that generate the observed series***
***2. predictforecast the future values of a series based on the history of series values."*** [6]

[6] Time Series Analysis: With Applications in R, (Springer Texts in Statistics) 2nd Edition, Cryer, Chan

# Time Series classification - I

- **univariate vs. multivariate:** Time series can have one or more variables that change over time.
- **discrete vs. continuous:** Based on whether time series observations are measured continuously or at discrete points in time.
- **linear vs. non-linear time-series models:** Based on whether its current value is a linear or non-linear function of past observations.

# Time Series classification - II

- **deterministic vs. non-deterministic:**
- *deterministic* time series can be expressed explicitly by an **closed form expression**. i.e. no random or probabilistic aspects (thus can be described exactly for all time in terms of a Taylor series).
- *non-deterministic* time series bear **random aspect** that prevents its behavior from being described explicitly (can be defined by statistical terms, i.e. by probability distributions and averages of various forms, such as means and variances). Due to information unavailability or the nature of the stochastic generating process.

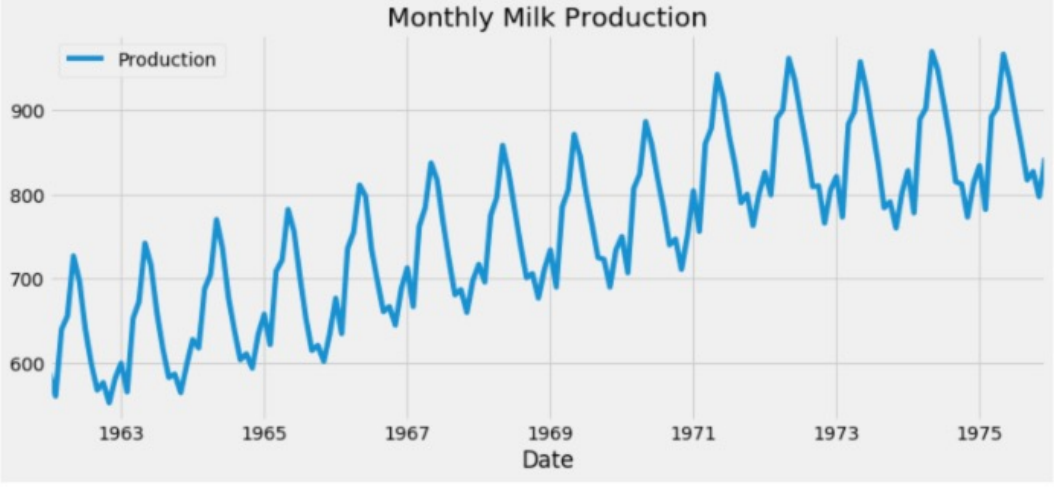
# Time Series classification - III

- **time-domain vs. frequency-domain models:**
- The *time-domain* approach **models future values as a function of past values and present values** - time series regression.
- *Frequency domain* models - time series represented as mixture of frequency signals - **Fourier representations**.

# Components of a Time Series

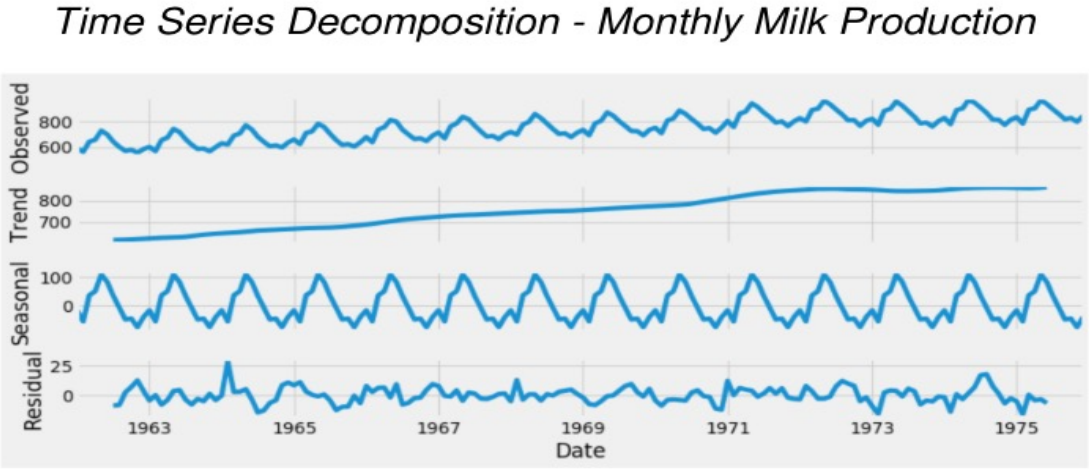
The first step in analyzing a time series for a predictive model is to identify the underlying pattern of the data over time.

*Milk Production in pounds per cow*



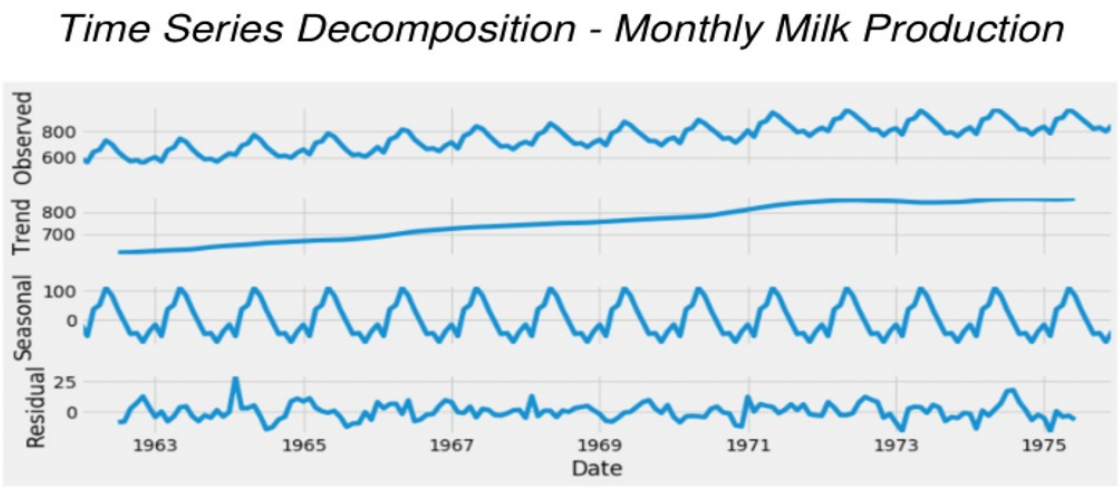

In general, a time series is affected by four components, i.e. **trend, seasonal, cyclical and irregular components**.

# Components of a Time Series




- **Trend :**
  The long-term gradual change in the series. This is the simplest trend pattern, as it demonstrates long-term growth or decline.
- **Seasonality:**
  Predictable, short-term patterns that occur within a single unit of time and repeat indefinitely.

# Components of a Time Series




- **Cyclical components**

Long-term swings in the data that may take years or decades to play out. These swings do not happen in a predictable manner and are often the result of external economic conditions.

- **Noise or irregular variation (residual)**

Random variation due to uncontrolled circumstances, which are not regular and also do not repeat in a particular pattern. These variations are caused by incidences such as war, strike, earthquake etc. There is no defined statistical technique for measuring random fluctuations in a time series

# Time Series Decomposition

Based on these four components, a time series can be decomposed using the following approaches. Decomposition is primarily used for time series analysis, and as an analysis tool it can be used to inform forecasting models on your problem.

1. **Additive Model:**
   - $Y(t) = T(t) + S(t) + I(t)$
   - These three components are independent of each other.
   - Assumption: seasonal variation is constant over time.
2. **Multiplicative Model:**
   - $Y(t) = T(t) \times S(t) \times I(t)$
   - These three components are not necessarily independent and they can affect one another.
   - Assumption: seasonal variation *increases* over time.

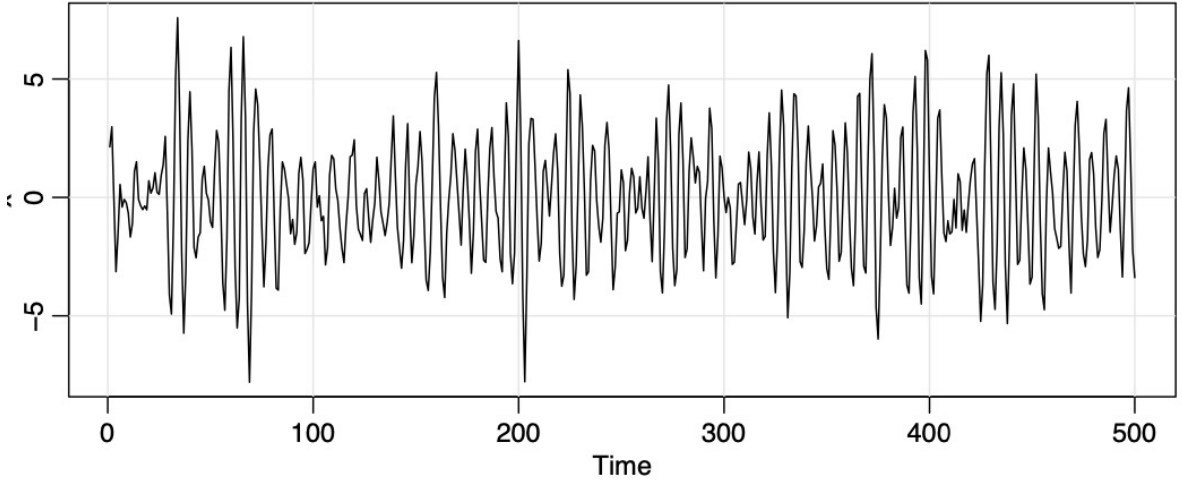
where $T(t)$: trend (with cycles), $S(t)$: seasonal, $I(t)$: irregular (random) components

# Time series as a stochastic process

- We can define a time series as a collection of random variables indexed based on the order they are obtained in time, $X_1, X_2, X_3, ...X_t$ will typically be discrete and vary over the integers $t = 0, \pm 1, \pm 2, ...$
- The collection of the random variables $X_t$ is referred to as a stochastic process, while the observed values are referred to as a realization of the stochastic process.
- A time series observed as **a collection of *n* random variables** at arbitrary time points $t_1, t_2, ..., t_n$, for any positive integer $n$, is provided by the joint distribution function, using the $n$ constants, $c_1, c_2, ..., c_n$ i.e.:
  - $F_{t_1,t_2,...t_n}(c_1, c_2, ..., c_n) = Pr(X_{t_1} \leqslant c_1, X_{t_2} \leqslant c_2, ..., X_{t_n} \leqslant c_n)$

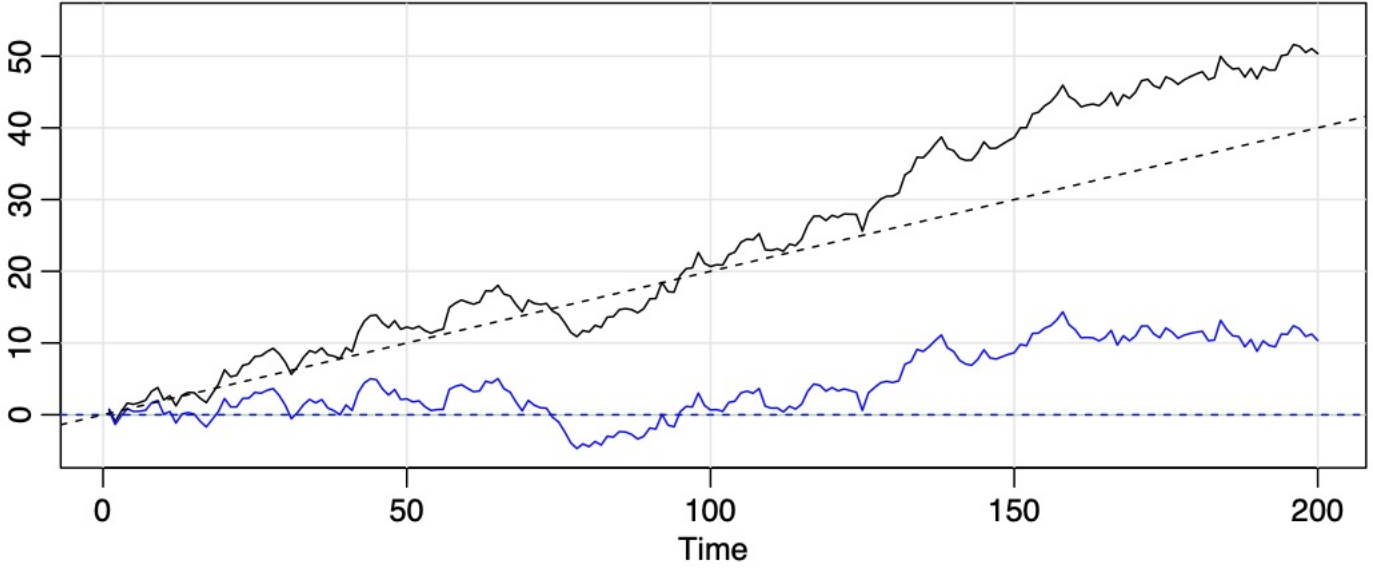
# Autoregressive time series

- Assume white noise series $w_t$ : $x_t = x_{t-1} - .9x_{t-2} + w_t$



# Random Walk with Drift time series

- *Time series model:* $x_t = \delta + x_{t-1} + w_t$ $t$ = 1, 2, . . ., $x_0$ = 0, and $w_t$ is white noise. δ: constant is called the *drift*
- when δ = 0, is called simply a *random walk*.

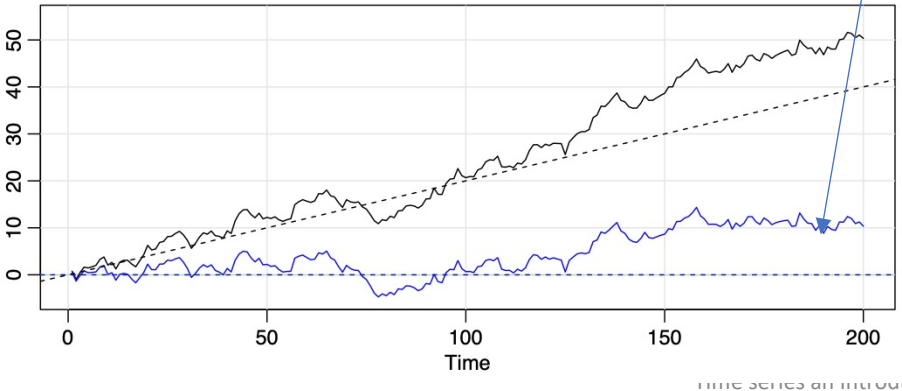


# Mean Function - Time Series

- The mean function is defined as $\mu_t = \mu_{X_t} = E[X_t] = -\int_{-\infty}^{\infty} x f_t(x) dx$, provided it exists, where $E$ the usual expected value operator.
- For White Noise: $\mu_{w_t} = E(w_t) = 0$
- For Random Walk:

$$\mu_{X_t} = E[X_t] = E[X_{t-1} + \delta + w_t] = E[\delta_t + w_t] = \delta_t + \sum_{i=1}^{t} E[w_i]$$

$$\Rightarrow \mu_{X_t} = \delta_t$$



# Autocovariance - Time Series

- **Autocovariance Function**
  - is defined as the second moment product
    $\gamma(s,t) = \gamma_X(s,t) = cov(X_s, X_t) = E[(X_s - \mu_s)(X_t - \mu_t)], \forall s,t$
  - $\gamma(s,t) = \gamma(t,s), \forall s,t$
- The autocovariance **measures the linear dependence between two points** on the same series observed at different times.

- **cross-covariance function** *between two series,* $x_t$ *and* $y_t$*, is*

$$\gamma_{xy}(s,t) = \mathrm{cov}(x_s, y_t) = \mathrm{E}[(x_s - \mu_{xs})(y_t - \mu_{yt})].$$

# Auto correlation - Time Series

- **Autocorrelation Function (ACF)**
  - $\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}$
  - From Cauchy-Schwarz inequality
    $|\gamma(s,t)|^2 \leqslant \gamma(s,s)\gamma(t,t) \Rightarrow -1 \leqslant \rho(s,t) \leqslant 1$
- ACF measures the **linear predictability of $X_t$ using only $X_s$**
  - If we can predict $X_t$ perfectly from $X_s$ through a linear relationship, then ACF will be either +1 or -1.
- *The* **cross-correlation function (CCF)** *between two series,* $x_t$ *and* $y_t$

$$\rho_{xy}(s,t) = \frac{\gamma_{xy}(s,t)}{\sqrt{\gamma_x(s,s)\gamma_y(t,t)}}.$$

# Stationary Time Series

- **strictly stationary** *time series: probabilistic behavior of every collection of values* $\{x_{t1},x_{t2},\ldots,x_{tk}\}$ *is identical to that of the time shifted set* $\{x_{t1+h},x_{t}$

$$\Pr\{x_{t_1} \leq c_1, \ldots, x_{t_k} \leq c_k\} = \Pr\{x_{t_1+h} \leq c_1, \ldots, x_{t_k+h} \leq c_k\}$$

- **weakly stationary** *time series*
  - The time series $X_t, t \in \mathbb{Z}$ is said to be weak stationary if:
    1. $E[x_t^2] < \infty, \forall t \in \mathbb{Z}$
    2. $E[x_t] = \mu, \forall t \in \mathbb{Z}$
    3. $\gamma_X(s,t) = \gamma_X(s+h, t+h), \forall s,t,h \in \mathbb{Z}$
- *mean value function,* $\mu_t$*, is constant and does not depend on time t*
- *autocovariance function, γ(s, t) depends on s and t only through their difference* $|s - t|$*.*
- Stationary datasets are those that have a *stable mean and variance*,

# Autocovariance and Autocorrelation for Stationary Time Series

- For the autocovariance $\gamma_X(s,t)$ of a stationary time series, we can use $s = t + h$ to take:

$$\gamma_X(s,t) = cov(X_{t+h}, X_t) = cov(X_h, X_0) = \gamma(h,0) = \gamma(h)$$

- The **Autocovariance of a Stationary Time Series** is:

$$\gamma(h) = cov(X_{t+h}, X_t) = E[(X_{t+h} - \mu)(X_t - \mu)]$$

- The **Autocorrelation of a Stationary Time Series** is:

$$\rho(s,t) = \frac{\gamma(t+h,t)}{\sqrt{\gamma(t+h,t+h)\gamma(t,t)}} = \frac{\gamma(h)}{\gamma(0)}$$

# Partial Autocorrelation Function (PACF)

- **PACF** gives the partial correlation of a stationary time series with its own lagged values, regressed the values of the time series at all shorter lags.
- equivalently, it is the autocorrelation between $X_t, X_{t+h}$ that is not accounted for by lags 1 through $h-1$, inclusive.
- It contrasts with the autocorrelation function, which does not control for other lags.
- It is defined as:
  - $\phi_{11} = corr(X_{t+1}, X_t) = \rho_1, h = 1$
  - $\phi_{hh} = corr(X_{t+h} - \hat{X}_{t+h}, X_{t+h} - \hat{X}_{t+h}), h \geqslant 2$
  - where $\hat{X}_t = \beta_1 X_{t-1} + \beta_2 X_{t-2} + ... + \beta_{h-1} X_{t+h-1}$
  - and $\hat{X}_{t+h} = \beta_1 X_{t+h-1} + \beta_2 X_{t+h-2} + ... + \beta_{h-1} X_{t+1}$

# AR Models

**Autoregressive Models (AR Models)** express the current value of the series, $X_t$ as a linear combination of $p$ past values, $X_{t-1}, X_{t-2}, ..., X_{t-p}$, including a random error in the approximation.

**Definition:** An autoregressive model of order p, **AR(p)**, is defined as

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + w_t \Rightarrow$$

$$\boxed{X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + w_t}$$

where $X_t$ is stationary, $\phi_1, \phi_2, ..., \phi_p$ are model parameters (where $p$ the length of historical points) and $w_t \sim wn(0, \sigma_w^2)$.

# AR Models: AR(0), AR(1), AR(p)

1. $AR(0)$ is the simplest AR process and has no dependence between the terms (white noise).
2. $AR(1)$ is defined as $X_t = \phi_1 X_{t-1} + w_t$
   - if $|\phi_1| \approx 0$ the process behaves like white noise.
   - if $\phi_1 = 1$ the process is equivalent to random walk with infinite variance (dependent on $t$, non-stationary).
   - if $\phi_1 < 0$ the process swings between positive and negative values.

# AR Models: Parameters Estimation

1. $p$ is a hyperparameter for the $AR(p)$ process, thus when fitting an AR(p) model p is known and we focus on estimating the coefficients $(\phi_1, \phi_2, ..., \phi_p)$.
2. Coefficients estimation can be done by different approaches:
   - Maximum Likelihood Estimation estimator (MLE).
   - Ordinary Least Squares estimator (OLS).

# Moving Average models

One problem of AR model is the ignorance of correlated noise structures in the time series. In other words, $w_t$ and its historical terms $w_{t-1}, w_{t-2}, \ldots, w_{t-q}$, include information that can be utilized for predictive models.

**Definition:** A Moving Average Model (MA) of order q, **MA(q)**, is defined as

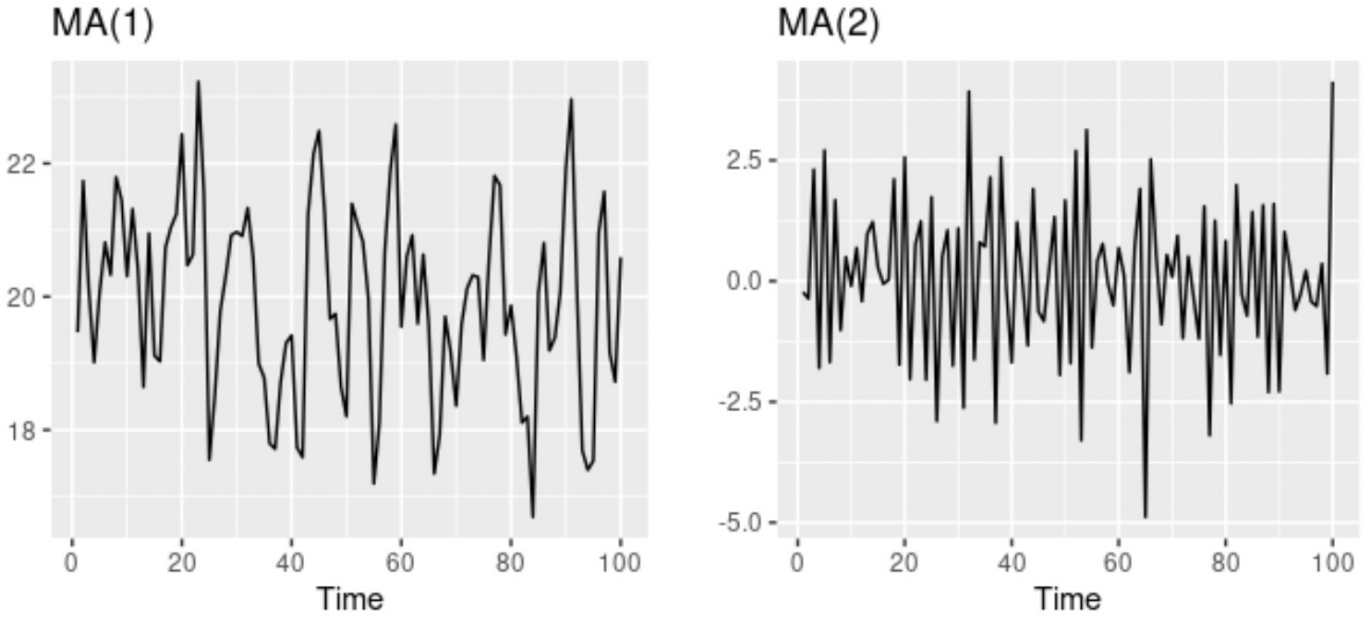
$$X_t = \theta_1 w_{t-1} + \theta_2 w_{t-2} + \ldots + \theta_q w_{t-q} + w_t \Rightarrow$$

$$\boxed{X_t = \sum_{j=1}^{q} \theta_j w_{t-j} + w_t}$$

where $X_t$ is stationary, $\theta_1, \theta_2, \ldots, \theta_p$ are model parameters (where $q$ the length of historical points) and $w_t \sim wn(0, \sigma_w^2)$.

- looks like a regression model, the difference is that the $w_t$ is not observable.
- In contrast with AR models, finite MA models are always stationary (observation is a weighted moving average over past forecast errors).

# MA(q) example




- MA(1) $y_t=20+\varepsilon_t+0.8\varepsilon_{t-1}$ MA(2) $y_t=\varepsilon_t-\varepsilon_{t-1}+0.8\varepsilon_{t-2}$
- $\varepsilon_t$: normally distributed white noise, mean: 0, variance =1

https://otexts.com/fpp2/MA.html

# ARMA Models

Autoregressive and Moving Average models can be combined and form **ARMA models**.

**Definition:** A stationary time series is **ARMA(p, q)** if

$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j}$$

where $\phi_p, \theta_q \neq 0$ and $w_t \sim wn(0, \sigma_w^2)$.

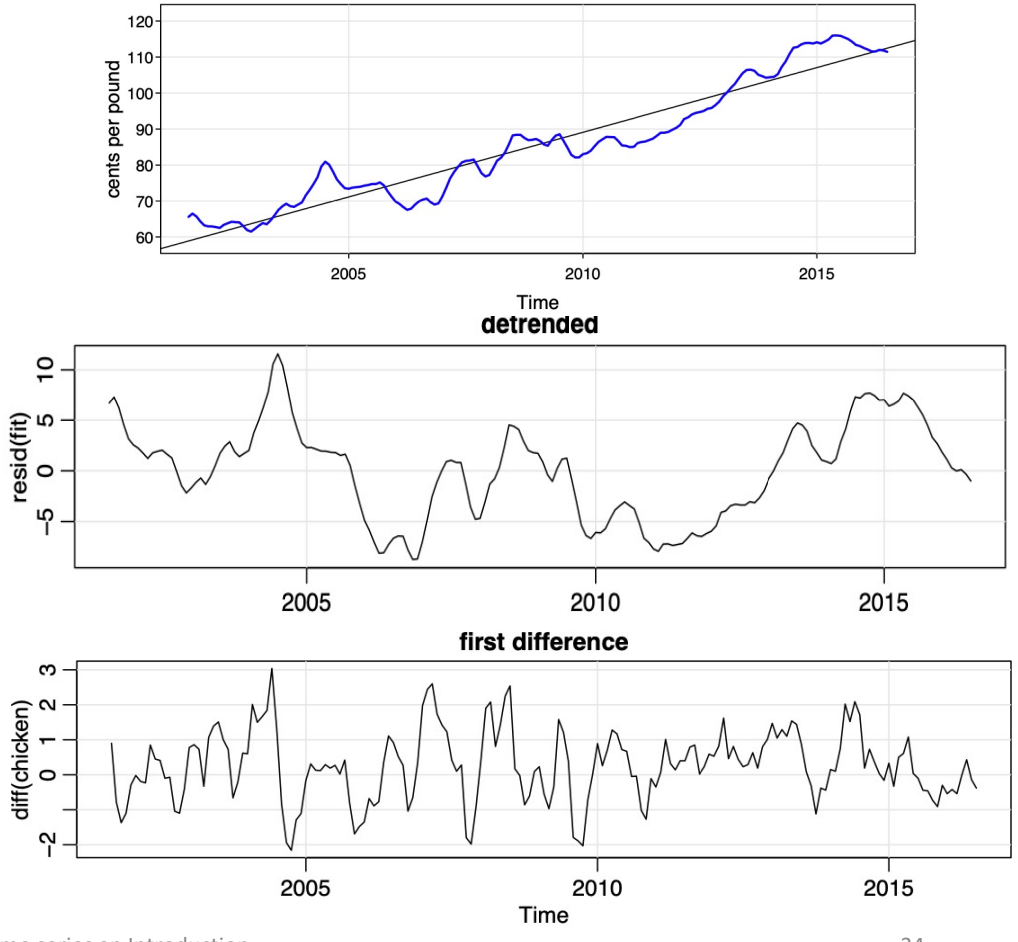
# Transforming Time Series to Stationary

prerequisite of ARMA models:
*stationarity of time series*

- Assume non-stationary time series: $x_t = \mu_t + y_t$
- $x_t$ are the observations, $\mu_t$ trend, $y_t$ is a stationary process.
- Remove the trend: estimate of the trend component: $\hat{\mu}_t$, and then work with the residuals

$$\hat{y}_t = x_t - \hat{\mu}_t$$

- Differencing: $x_t = x_t - x_{t-1}$



# VAR Models

## Limitation of univariate models

One limitation of the models that we have considered so far is that they impose a unidirectional relationship — the forecast variable is influenced by the predictor variables, but not vice versa. However, there are many cases where variables affect each other.

- → **Vector autoregressive (VAR) model** can extend the univariate models and overcome this limitation.
- → Forecasts are generated from a VAR in a recursive manner. The VAR **generates forecasts for each variable** included in the system.

# VAR Models

- Feedback relationships between variables are allowed for in the Vector autoregressive (VAR) framework. In this framework, **all variables are treated symmetrically**. They are modelled as if they all influence each other equally. In formal terminology, variables are now treated as “endogenous”.
- VAR models **generalize the single-variable (univariate) autoregressive model** by allowing for multivariate time series.
- In practice a VAR is a **system regression model** that treats all the variables as endogenous and allows each of them to depend on *p* lagged values of itself and of all the other variables in the system.
- VAR models are a specific case of more general VARMA models. VARMA models for multivariate time series include the VAR structure above along with moving average terms for each variable.

# VAR Models

**Vector autoregression (VAR)** is a statistical model used to capture the relationship between multiple quantities as they change over time.

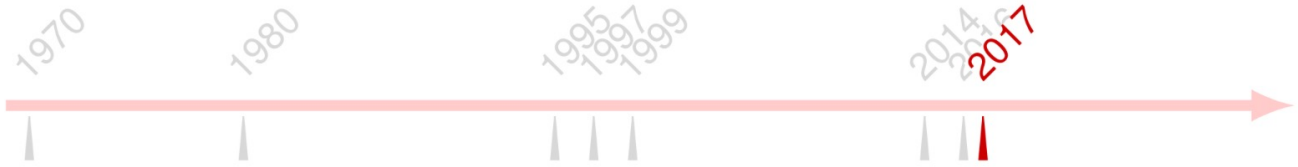
**Definition:** A VAR model of order p, **VAR(p)**, is a process that can be represented as:

$$\boldsymbol{y}_t = \boldsymbol{a}_0 + \boldsymbol{A}_1 \boldsymbol{y}_{t-1} + \boldsymbol{A}_2 \boldsymbol{y}_{t-2} + \boldsymbol{A}_p \boldsymbol{y}_{t-p} + \boldsymbol{u}_t \Rightarrow$$

$$\boxed{\boldsymbol{y}_t = \boldsymbol{a}_0 + \sum_{i=1}^{p} \boldsymbol{A}_i \boldsymbol{y}_{t-i} + \boldsymbol{u}_t}$$

where $\boldsymbol{y}_t$ is a $N \times 1$ vector containing $N$ endogenous variables, $\boldsymbol{a}_0$ is a $N \times 1$ vector of constants, $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_p$ are the $p$ $N \times N$ matrices of autoregressive coefficients and $\boldsymbol{u}_t$ is a $N \times 1$ vector of white noise disturbances.

# History of Time Series




- early 1970s: ARIMA
- late 1970s: Bayesian Approach
- 1980s: GARCH
- 1995: Random Forest
- 1997: Support Vector Regression
- 1999: Boosted Decision Trees
- 2014: LSTMs, GRUs, Seq2Seq
- 2016: Dilated CNNs
- 2017: Transformer

## Limitations of Statistical Methods

Focus on

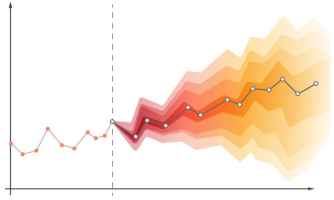
- complete data
- linear relationships
- fixed temporal dependencies
- univariate data
- 1-step forecasts

## Why use Deep Learning for Time Series?

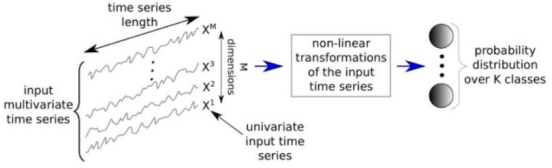
- Learns patterns in data with complex non-linear dependencies
- Supports multiple inputs and outputs
- Has shown good performance in many scenarios
- Models are flexible and expressive
- Can be trained with large datasets
- Easy to introduce exogenous features into the model

# Different tasks in Time series

**Forecasting**
(point or probabilistic)



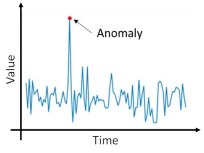
**Classification**
(to identify a time series as coming from one of possibly many sources or predefined groups)




Other tasks:

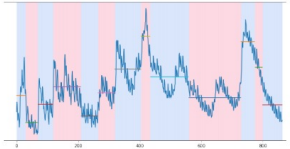
1. Anomaly Detection

*Find abnormal events in a time series*



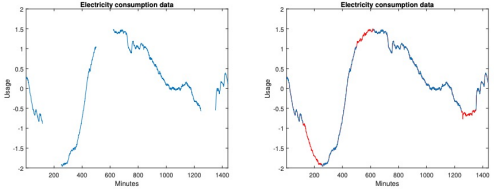

2. Segmentation/ change-point detection

*Find significant abrupt changes in the time series*



3. Completion/interpolation

*Recover missing/lost samples in a time series*




4. Query by content/indexation

*Given an input time series, retrieve the closest time series in a large database up to a given measure of fit*

# DL Architectures for Time Series Forecasting

1-step-ahead forecasting:

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \boldsymbol{x}_{i,t-k:t}, \boldsymbol{s}_i)$$

- $\hat{y}_{i,t+1}$: Model forecast
- $y_{i,t-k:t}$: observations of target over a look-back window k
- $\boldsymbol{x}_{i,t-k:t}$: enternal inputs over a look-back window k
- $\boldsymbol{s}_i$: static metadata associated with the entity (e.g. sensor location)

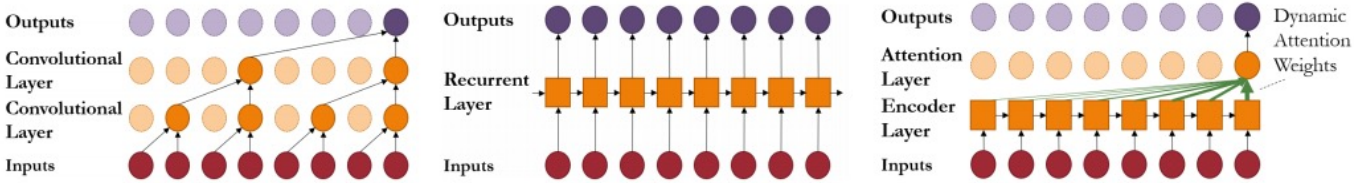
multi-horizon forecasting:

$$\hat{y}_{t+\tau} = f(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{u}_{t-k:t+\tau}, \boldsymbol{s}, \tau)$$

- $\tau$: τ is a discrete forecast horizon
- $\boldsymbol{u}_{t-k:t+\tau}$: known future inputs (e.g. date information, such as the day-of-week or month) across the entire horizon

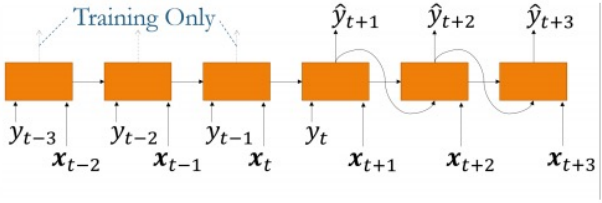
Time Series Forecasting With Deep Learning: A Survey, Bryan Lim and Stefan Zohren, https://arxiv.org/pdf/2004.13408.pdf

# Basic Building Blocks

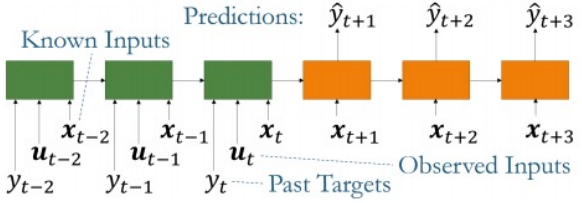



(a) CNN Model. (b) RNN Model. (c) Attention-based Model.

## Multi-horizon Forecasting Models




(a) Iterative Methods




(b) Direct Methods

autoregressive deep learning architectures, produce multi-horizon forecasts by recursively feeding samples of the target into future time steps

→ large error accumulations

produce forecasts directly using all available inputs, make use of sequence-to-sequence architectures, (an encoder summarises past information and a decoder combines them with known future inputs)

→ Fixed maximum forecast horizon

Time Series Forecasting With Deep Learning: A Survey, Bryan Lim and Stefan Zohren, https://arxiv.org/pdf/2004.13408.pdf

# Outputs and Loss functions

1. **Point Estimates:** determine the expected value of a future target: reformulate problem to a classification task for discrete outputs (e.g. forecasting future events), and regression task for continuous outputs.

$$\mathcal{L}_{classification} = -\frac{1}{T}\sum_{t=1}^{T} y_t \log(\hat{y}_t) + (1 - y_t)\log(1 - \hat{y}_t)$$

$$\mathcal{L}_{regression} = \frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

2. **Probabilistic Outputs:** model uncertainties, use deep neural networks to generate parameters of known distributions. e.g. Gaussian to forecast continuous targets

networks output: mean, variance parameters for the predictive distributions / step:

$$y_{t+\tau} \sim N(\mu(t,\tau), \zeta(t,\tau)^2),$$

$$\mu(t,\tau) = \boldsymbol{W}_\mu \boldsymbol{h}_t^L + \boldsymbol{b}_\mu,$$

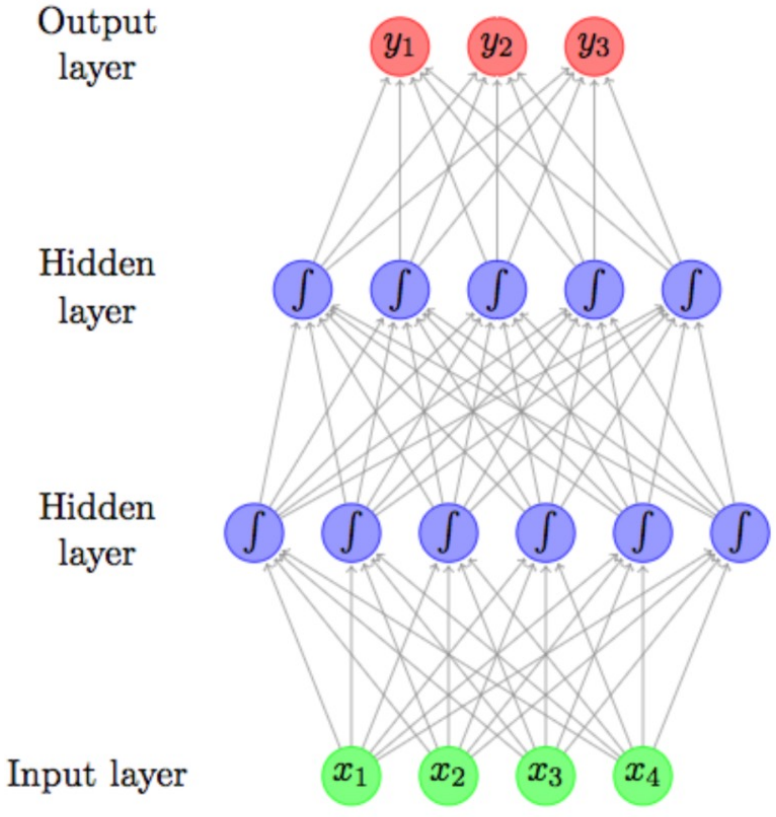
Final layer of the network

$$\zeta(t,\tau) = \text{softplus}(\boldsymbol{W}_\Sigma \boldsymbol{h}_t^L + \boldsymbol{b}_\Sigma),$$

softplus activation: ensure variance takes only positive values

Time Series Forecasting With Deep Learning: A Survey, Bryan Lim and Stefan Zohren, https://arxiv.org/pdf/2004.13408.pdf

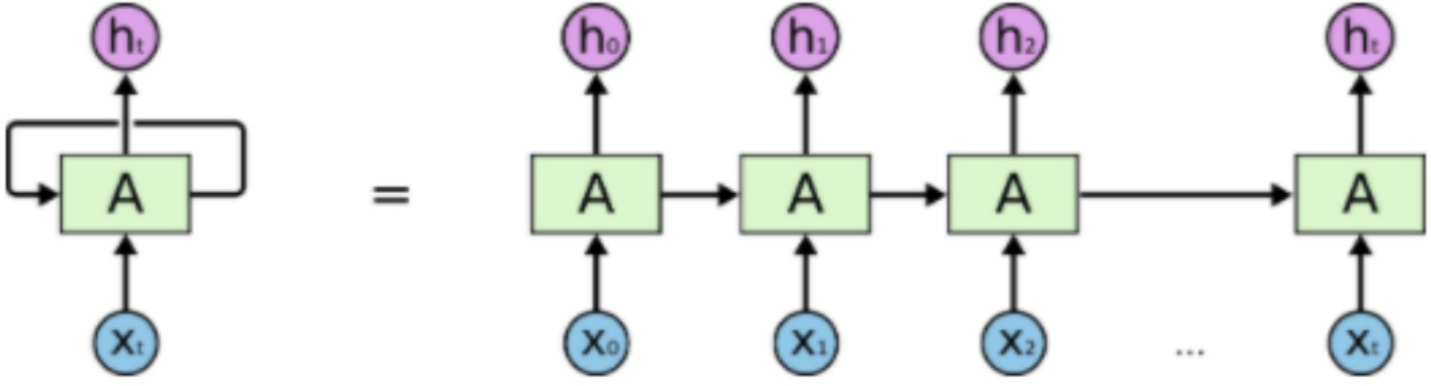
# MLPs




$$NN_{MLP2}(\mathbf{x}) = y$$
$$\mathbf{h}^1 = g^1(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$
$$\mathbf{h}^2 = g^2(\mathbf{h}^1\mathbf{W}^2 + \mathbf{b}^2)$$
$$\mathbf{y} = \mathbf{h}^2\mathbf{W}^3$$

# Recurrent Neural Networks

- RNNs model the order of the data explicitly
- This chain-like nature reveals that recurrent neural networks are intimately related to sequences
- Has been the most common choice for sequence modelling



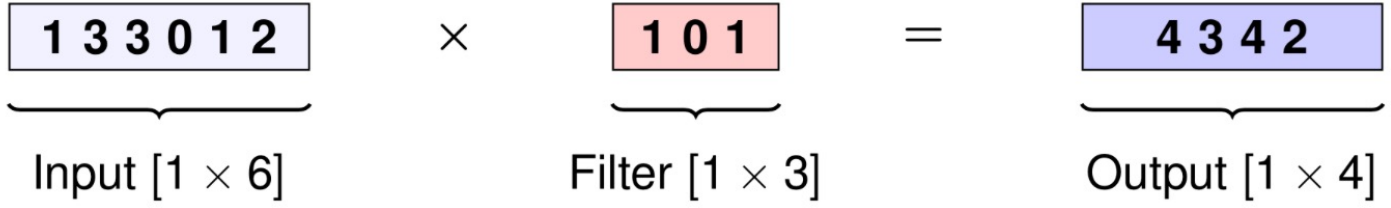

An unrolled Recurrent Neural Network [7]

[7] *source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Convolutional Neural Networks

- Convolution layer
  - units in the hidden layer operate on a field of the output
  - weights are shared across input
- Motivation for CNNs in sequence learning
  - Success of CNN in CV and recently in NLP
  - Achieves state-of-the-art accuracy in audio synthesis and machine translation
- Main advantages:
  - Lower level of model complexity than RNNs
  - Parallelization of computations
  - Dilated CNN can outperform RNN in sequence modelling
- 2D Convolutions encode spatial invariance
- 1D Convolutions encode temporal invariance, “stationarity”

# 1D Convolutions




1D Convolution with 1 filter ($stride = 1$), $t = 6$, $\#features = 1$

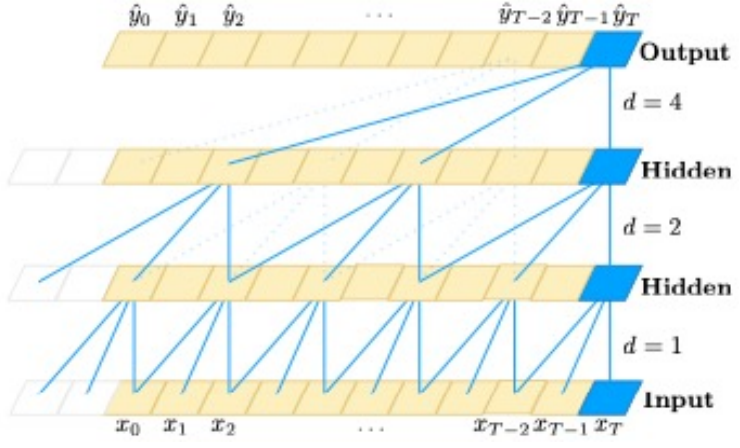
- A convolution kernel that is convolved with the input over a single temporal dimension
- Filters are trained to detect features in the sequence, regardless of where they appear
- Multiple filters can detect multiple features
- Can be implemented to multivariate time series (Input [$t \times \#features$])

# Causality

- *causal constraint*: $y_t$ depends only on $x_0,...,x_t$ and not on any "future" inputs $x_{t+1},...,x_T$.
- Learning: network *f* that minimizes some expected loss between the actual outputs and the predictions, $L(y_0,...,y_T,f(x_0,...,x_T))$
- This formalism encompasses settings
  - auto- regressive prediction: predict some signal given its past:
    - setting the target output to be simply the input shifted by one time step.
  - does not capture domains (machine translation, or sequence- to-sequence prediction)
    - the entire input sequence (including *"future" states*) can be used to predict each output

# Causal Convolution

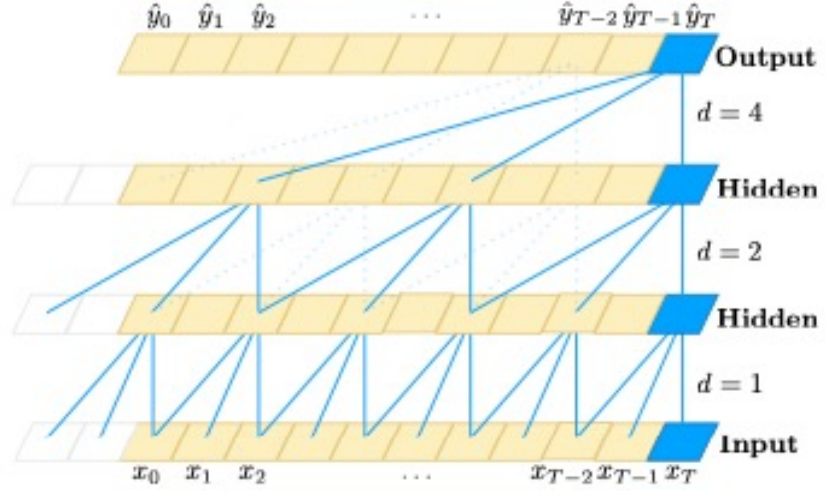
- Temporal Convolution Networks (TCN) constraints:
  - network produces an output of the same length as the input
    - To accomplish this: uses a 1D fully-convolutional network (FCN) architecture
    - each hidden layer is the same length as the input layer, and zero padding of length (kernel size − 1) is added to keep subsequent layers the same length as previous ones.
  - there can be no leakage from the future into the past.
    - TCN uses *causal convolutions:* output at time t is convolved **only** with elements <t in the previous layer.
  - *TCN = 1D FCN + causal convolutions*




An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

# Causal Convolution

- Disadvantage:  in order to achieve a long effective history size need large kernel size -  extremely deep network or very large filters
- *Dilated convolution*: to allow for both very deep networks and very long effective history



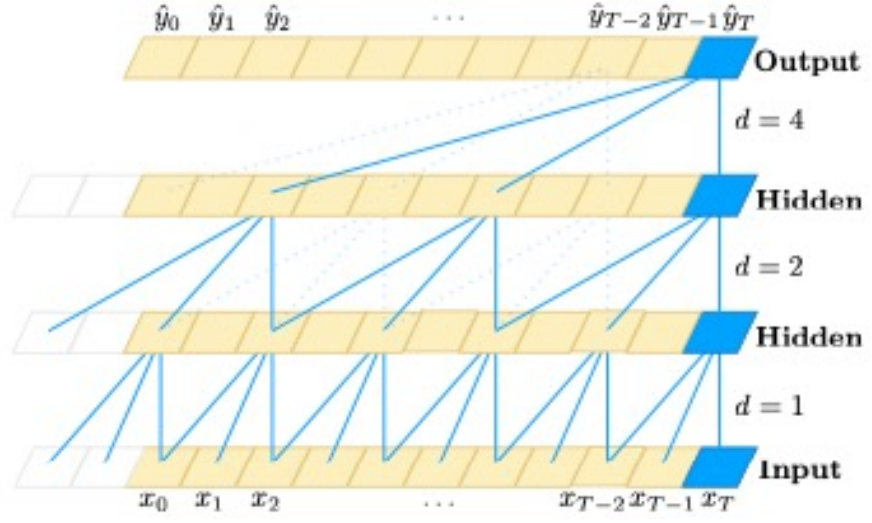

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun,  https://arxiv.org/pdf/1803.01271.pdf

# Dilated Convolutions

- 1-D sequence input $x \in R^n$ and a filter $f:\{0,...,k-1\} \rightarrow R$,
- Dilated convolution operation F on element s of the sequence:

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}$$

- *d*: dilation factor, k: filter size, $(s - d \cdot i)$ direction of the past.
- is equivalent to introducing a fixed step between every two adjacent filters.
- d = 1: regular convolution
- d>1: enables output at the top level to represent a wider range of inputs: expanding the receptive field of a ConvNet.

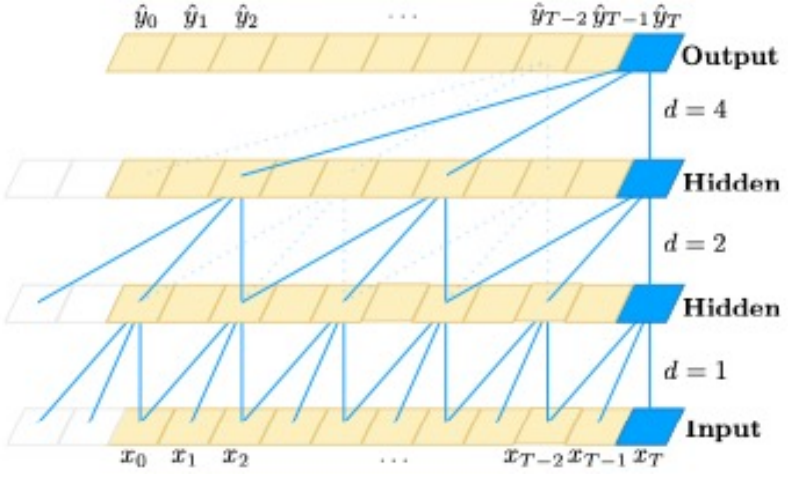


# Dilated Convolutions

Longer memory - increase the receptive field of the TCN:
- choosing larger filter sizes *k*
- increasing the dilation factor *d*:
- effective history for this layer *(k − 1)d*.

In dilated convolutions, increase d exponentially with the depth of the network: $d = O(2^i)$. *i:* level in the network. Ensures

- each input treated by some filter in the effective history,
- allowing for an extremely large effective history using deep networks.




An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

# Dilated Convolutions

Advantages TCNs for sequence modelling.

- *Parallelism*. Unlike in RNNs, convolutions can be done in parallel since the same filter is used in each layer.
- *Flexible receptive field size*: stacking more dilated (causal) convolutional layers, using larger dilation factors, increasing the filter size. better control of the model's memory size, and are easy to adapt to different domains.
- *Stable gradients*. Unlike recurrent architectures, TCN has a backpropagation path different from the temporal direction of the sequence. Thus avoids the problem of exploding/vanishing gradients
- *Low memory requirement for training*. For long input sequence, LSTMs and GRUs use a lot of memory for partial results for their multiple cell gates. TCN filters are shared across a layer, with the backpropagation path depending only on network depth.
- *Variable length inputs*. like RNNs, TCNs take in inputs of arbitrary lengths by sliding the 1D convolutional kernels.

Disadvantages

- *Data storage during evaluation*, RNNs only need hidden state $h_t$ and the current input $x_t$ in order to generate a prediction. In contrast, TCNs need the raw sequence up to the effective history length
- Potential parameter change for a transfer of domain.

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

# Experimental Evaluation – Data Sets

- **The adding problem**: each input consists of a length-n sequence of depth 2, with all values randomly chosen in [0, 1], and the second dimension being all zeros except for two elements that are marked by 1. The objective is to sum the two random values whose second dimensions are marked by 1. Simply predicting the sum to be 1 should give an MSE of about 0.1767. Long discussed data set
- **Sequential MNIST and P-MNIST**: test a recurrent network's ability to retain information from the distant past. MNIST images presented to the model as a 784 × 1 sequence for digit classification. P-MNIST, the order of the sequence is permuted at random.
- **Copy memory**: each input sequence has length T + 20. The first 10 values chosen randomly in [ 1 . . 8], the rest being all 0, except for the last 11 entries that are filled with the digit '9' (the first '9' is a delimiter). The goal is to generate an output of the same length that is zero everywhere except the last 10 values after the delimiter, where the model is expected to repeat the 10 values it encountered at the start of the input. (25)1130403028000099999999999 => (25)00000000000000001130403028
- **JSB Chorales and Nottingham**: is a polyphonic music dataset 382 harmonized chorales by J. S. Bach. input is a sequence of elements (88-bit binary code for the 88 keys on a piano, 1: key pressed at a given time. Nottingham: polyphonic music dataset ~ 1,200 folk tunes. Used for recurrent sequence modelling - performance: negative log-likelihood (NLL).
- **PennTreebank**. Character-level and word-level language modeling. Character PTB contains 5,059K characters for training, 396K for validation, and 446K for testing, with an alphabet size of 50. As word-level language corpus, 888K words training, 70K validation, and 79K testing, with a vocabulary size of 10K.
- **Wikitext-103**. 110 times as large as PTB, vocabulary size ~ 268K – dataset: 28K Wikipedia articles (~ 103 million words) training, 60 articles (~ 218K words) validation, 60 articles (246K words) testing.
- **LAMBADA**. 10K passages extracted from novels, average of 4.6 sentences as context, and 1 target sentence the last word of which is to be predicted. Oriented to capturing information from longer and broader context. training data full text of 2,662 novels, >200M words. The vocabulary size ~93K.
- **text8**. character-level language modeling (Mikolov et al., 2012). text8 is about 20 times larger than PTB, with about 100M characters from Wikipedia (90M for training, 5M for validation, and 5M for testing). The corpus contains 27 unique alphabets.

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

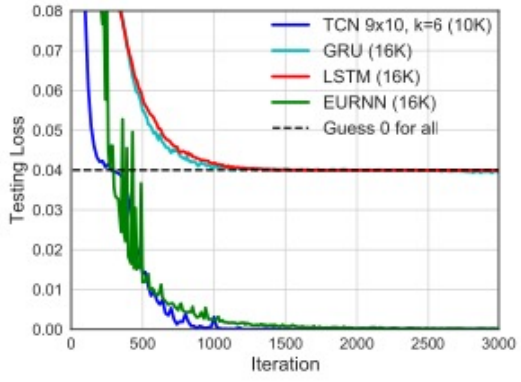
# Experimental Evaluation

| Sequence Modeling Task | Model Size (≈) | 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** |

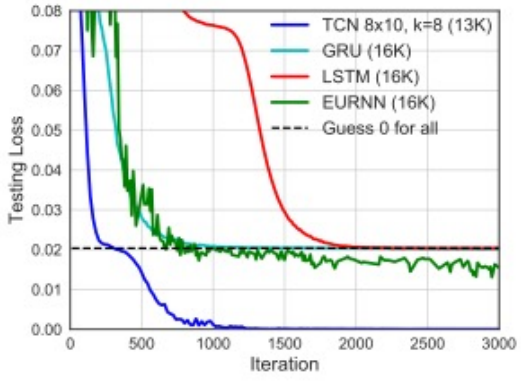
An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

# Experimental Evaluation

- Copy memory task



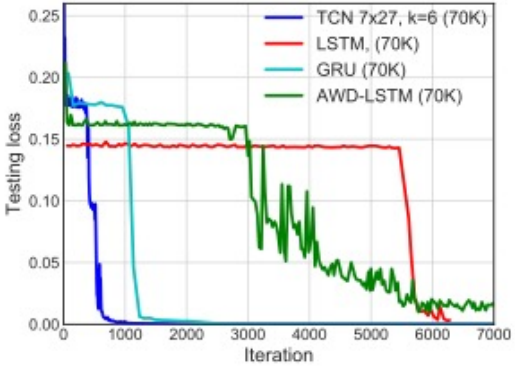

(a) $T = 500$



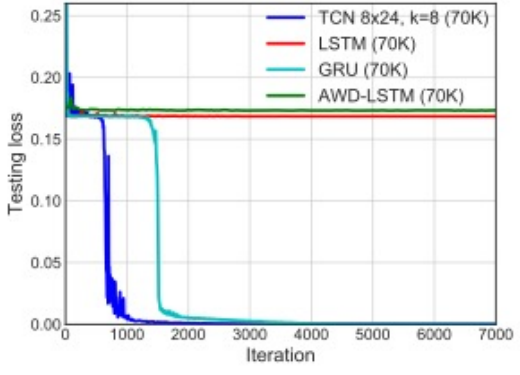

(b) $T = 1000$

- adding problem task

different sequence length



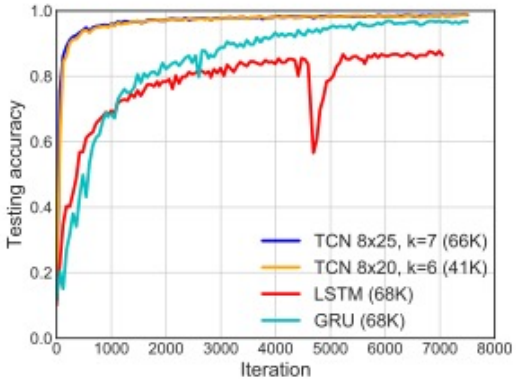

(a) $T = 200$



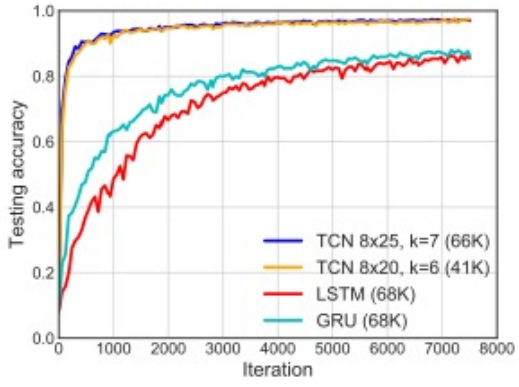

(b) $T = 600$

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, Shaojie Bai. J. Zico Kolter, Vladlen Koltun, https://arxiv.org/pdf/1803.01271.pdf

# Experimental Evaluation

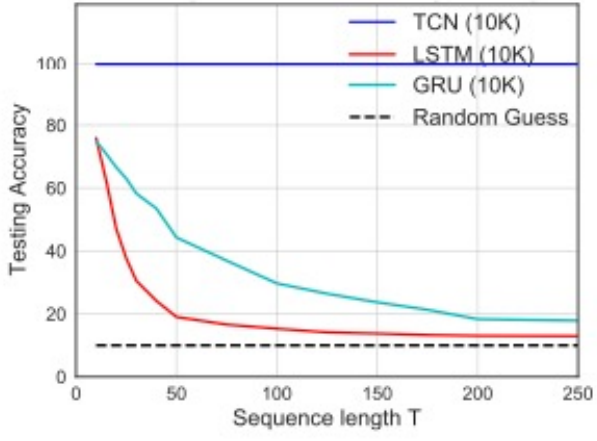



(a) Sequential MNIST




(b) P-MNIST

- Accuracy (copy memory task) for sequences of different lengths T



# Transformer model for Forecasting [1]

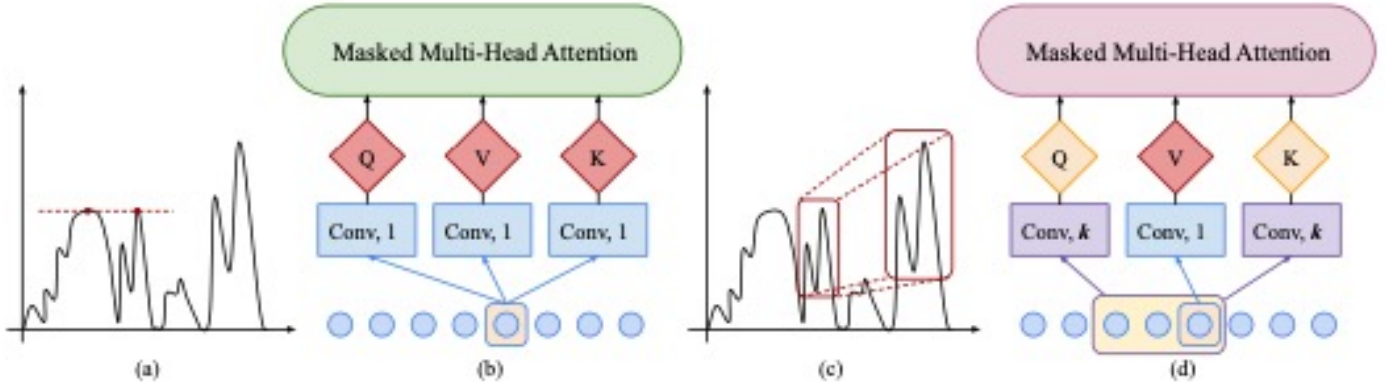
- Use of transformers in NLP is impressive
- For time series and sequences two major weaknesses:
  - *locality-agnostics*: the dot- product self-attention in vanilla Transformer is insensitive to local context - prone to anomalies in time series;
  - *memory bottleneck*: space complexity of Transformer $\Theta(L^2)$ L: sequence length – thus modelling long time series infeasible.

[1] proposed

- convolutional self-attention by producing queries and keys with *causal convolution* so that local context can be better incorporated into attention mechanism.
- *LogSparse* Transformer with only $\Theta(L(\log L)^2)$ memory cost
- improving *forecasting accuracy* for time series with fine granularity
- strong long-term dependencies under constrained memory budget.
- experiments on synthetic data and real- world datasets show that it compares favorably to the state-of-the-art.

[1]"Enhancing the Locality and Breaking the Memory Bottleneck of Transformer onTime Series Forecasting, Li et al., 2019

# Transformer model for Forecasting




vanilla vs. convolutional self-attention layers.

- “Conv, 1” and “Conv, k”: kernel size {1, k} with stride 1
- Vanilla self-attention in (b), may wrongly match point-wise inputs as in (a).
- Convolutional self-attention (d), convolutional layers of kernel size k with stride 1 to transform inputs (with proper paddings) into queries/keys.
- Thus locality awareness can correctly match the most relevant features based on shape matching in (c).

# Transformer model for Forecasting

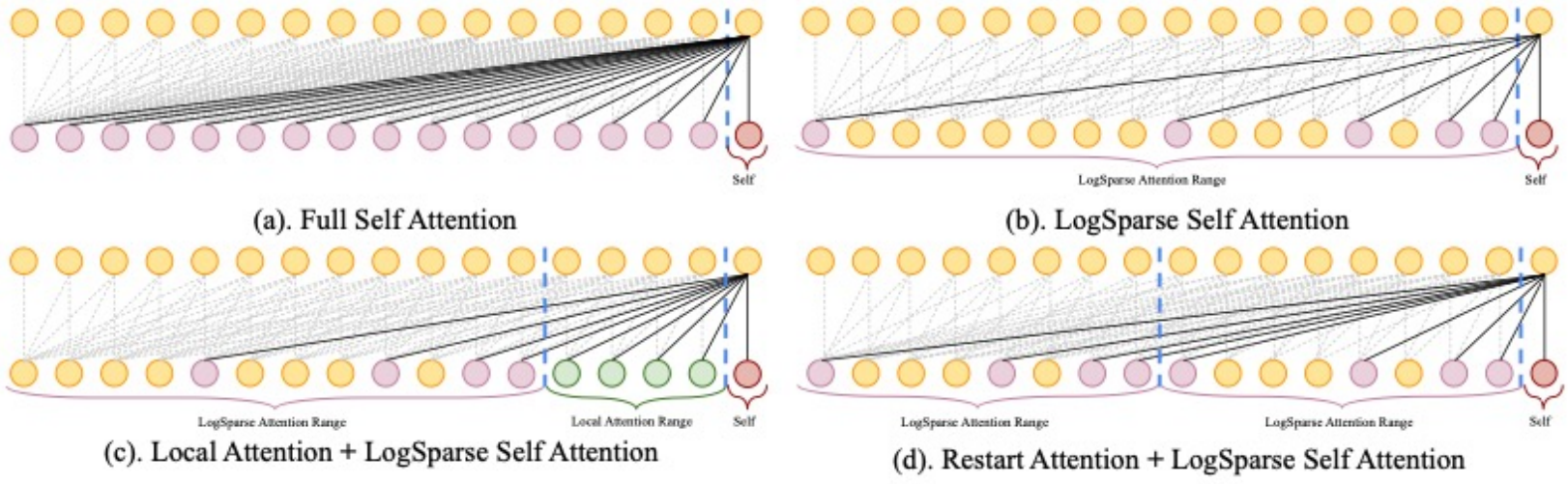



Illustration of different attention mechanism between adjacent layers in Transformer.

- Local Attention: allow each cell to densely attend to cells in its left window of size $O(\log_2 L)$ so that more local information/trend, can be leveraged for current step forecasting. Beyond the neighbor cells, can resume *LogSparse* attention strategy Figure 3(c).
- Restart Attention: divide the input length L into subsequences and set each subsequence length $L_{sub} \sim L$. For each of them, we apply the *LogSparse* attention strategy - Figure 3(d).

# Experimental Evualuation

- Data Sets
  - **electricity-f (fine):** electricity consumption of 370 customers recorded every 15 minutes
  - **electricity-c (coarse) :** aggregated electricity-f by every 4 points, producing hourly electricity consumption.
  - **traffic-f (fine)**: occupancy rates of 963 freeway in San Francisco recorded /20 minutes
  - **traffic-c (coarse)**:  hourly occupancy averaging every 3 points in traffic-f.
  - **solar dataset6**: solar power production records January to August 2006, sampled /hour from 137 PV plants in Alabama.
  - **wind7:** contains daily estimates of 28 countries' energy potential (1986 – 2015) as % of a power plant's maximum output.
  - **M4-Hourly**: 414 hourly time series from M4 competition

# Experimental Evualuation

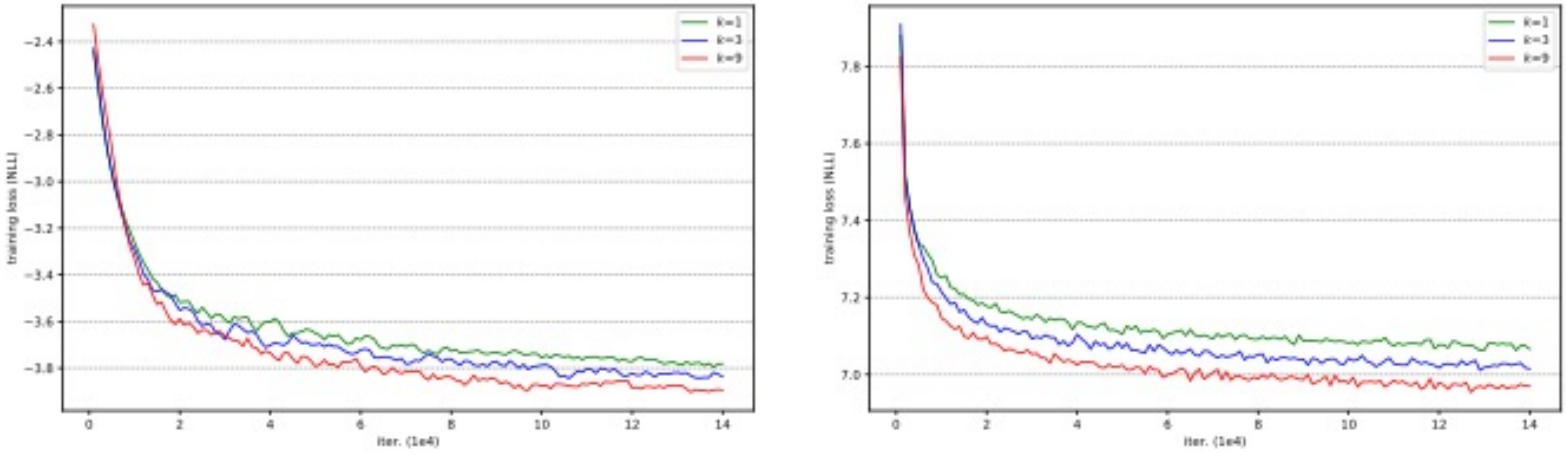
| | ARIMA | ETS | TRMF | DeepAR | DeepState | Ours |
|---|---|---|---|---|---|---|
| e-$c_{1d}$ | 0.154/0.102 | 0.101/0.077 | 0.084/- | 0.075◇/0.040◇ | 0.083◇/0.056◇ | **0.059/0.034** |
| e-$c_{7d}$ | 0.283◇/0.109◇ | 0.121◇/0.101◇ | 0.087/- | 0.082/0.053 | 0.085◇/0.052◇ | **0.070/0.044** |
| t-$c_{1d}$ | 0.223/0.137 | 0.236/0.148 | 0.186/- | 0.161◇/0.099◇ | 0.167◇/0.113◇ | **0.122/0.081** |
| t-$c_{7d}$ | 0.492◇/0.280◇ | 0.509◇/0.529◇ | 0.202/- | 0.179/0.105 | 0.168◇/0.114◇ | **0.139/0.094** |

- R0.5/R0.9-loss – from ρ-quantile loss:

$$R_\rho(\mathbf{x}, \hat{\mathbf{x}}) = \frac{2 \sum_{i,t} D_\rho(x_t^{(i)}, \hat{x}_t^{(i)})}{\sum_{i,t} |x_t^{(i)}|}, \quad D_\rho(x, \hat{x}) = (\rho - \mathbf{I}_{\{x \leq \hat{x}\}})(x - \hat{x})$$

- x^ is the empirical ρ-quantile of the predictive distribution and $I\{x \leq x\hat{}\}$ is an indicator function, $\varrho \in (0, 1)$

# Experimental Evualuation




- Training loss curve comparison with kernel size k ∈ {1, 3, 9} traffic-c (left) and electricity-c (right) dataset.
- Being aware of larger local context size, the model can achieve lower training error and converge faster.

Table 2: Average $R_{0.5}/R_{0.9}$-loss of different kernel sizes for rolling-day prediction of 7 days.

| | $k = 1$ | $k = 2$ | $k = 3$ | $k = 6$ | $k = 9$ |
|---|---|---|---|---|---|
| electricity-c$_{1d}$ | 0.060/**0.030** | 0.058/**0.030** | **0.057**/0.031 | **0.057**/0.031 | 0.059/0.034 |
| traffic-c$_{1d}$ | 0.134/0.089 | 0.124/0.085 | 0.123/0.083 | 0.123/0.083 | **0.122/0.081** |

# Experimental Evualuation

## sparse attention vs. full attention models

- Average *R0.5/R0.9-loss* comparisons, rolling-day prediction 7 days, kernel k = 6.
- with/without convolutional self-attention "Full" means models are trained with full attention while "Sparse" means they are trained with our sparse attention strategy.
- "+ Conv" means models are equipped with convolutional self-attention with

| Constraint | Dataset | Full | Sparse | Full + Conv | Sparse + Conv |
|---|---|---|---|---|---|
| Memory | electricity-$f_{1d}$ | 0.083/0.051 | 0.084/**0.047** | **0.078**/0.048 | 0.079/0.049 |
| | traffic-$f_{1d}$ | 0.161/0.109 | 0.150/0.098 | 0.149/0.102 | **0.138/0.092** |
| Length | electricity-$f_{1d}$ | 0.082/0.047 | 0.084/0.047 | **0.074/0.042** | 0.079/0.049 |
| | traffic-$f_{1d}$ | 0.147/0.096 | 0.150/0.098 | 0.139/**0.090** | **0.138**/0.092 |

- R0.5/R0.9-loss of datasets with various granularities.
- subscript of each dataset forecasting horizon (days).

| | electricity-$f_{1d}$ | traffic-$f_{1d}$ | solar$_{1d}$ | M4-Hourly$_{2d}$ | wind$_{30d}$ |
|---|---|---|---|---|---|
| TRMF | 0.094/- | 0.213/- | 0.241/- | -/- | 0.311/- |
| DeepAR | 0.082/0.063 | 0.230/0.150 | 0.222/0.093 | 0.090°/0.030° | 0.286/0.116 |
| Ours | **0.074/0.042** | **0.139/0.090** | **0.210 /0.082** | **0.067 /0.025** | **0.284/0.108** |

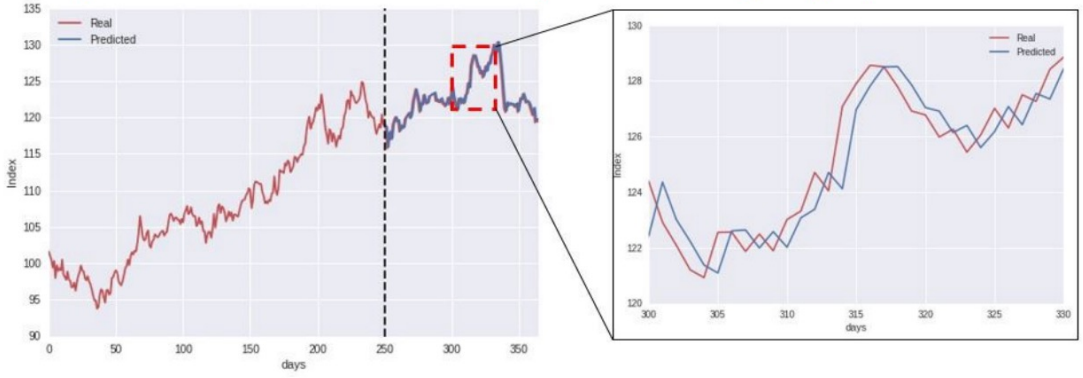
# Is DL the optimal solution?

DL approaches are flexible and achieve state-of-the-art results for several tasks BUT…

- Good performances does NOT mean good understanding of the data/good results
- DL when used as a black box (without understanding of the data/their properties/models) is not enough, since experts cannot interpret the results

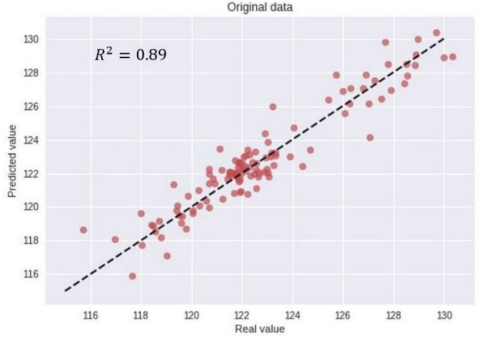
# An example where DL fails

Blog post from V. Flovik
https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424




Prediction of stock index with an LSTM. Predictions seem great!

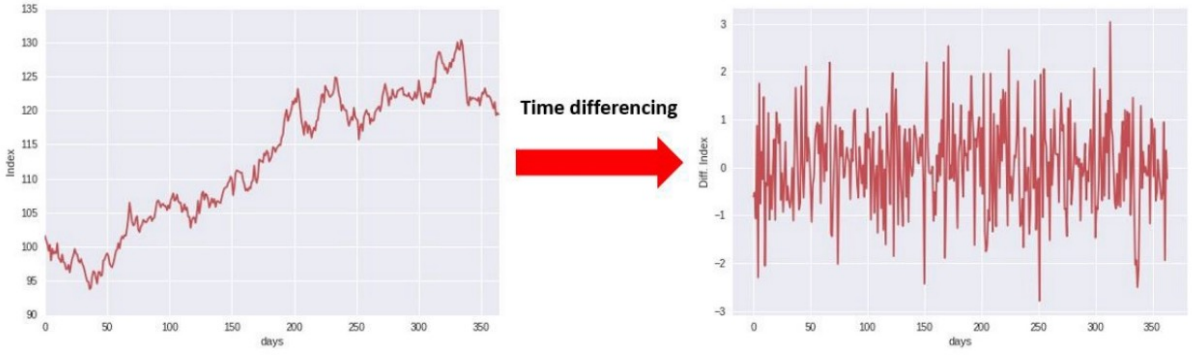
In fact, the value of the previous time-step is copied!




Even while copying the previous step, we achieved great metrics, such as RMSE.

This should not be considered as a good forecast!

# An example where DL fails




Time differencing: $x_t$-$x_{t-1}$
In fact, the series was a random walk, impossible to predict.
This could have been detected with some pre-processing steps.

**There is a need to:**

1. Examine and preprocess the data
2. Build a strong model for temporal pattern extraction
3. Choose the right metrics for learning (e.g. the standard mse is sensitive to outliers) and the evaluation of predictions

# Conclusion-
# Hidden tasks for time series learning

1. **Understand the data:** know where they come from, how they were obtained, their characteristics, interact with domain-experts and understand their problems
2. **Improve the data:** consolidate the data (*denoising, detrending, detection/removal of outliers*), transform input data to more meaningful representations (i.e. graphs).
3. **Extract information from the data:** find repetitive patterns, change-points, anomalies
4. **Model the data:** powerful but simple, adaptive and interpretable models

# THANK YOU!

Acknowledgement:
Sissy Kosma (PhD student)

# Refererences


- 25 years of time series forecasting, Jan G. De Gooijer, Rob J. Hyndman b, 2006
- Deep Learning for Time-Series Analysis, 2017: https://arxiv.org/abs/1701.01887
- Deep Learning for Time-Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python, Jason Brownlee, Machine Learning Mastery, 2018
- FORECASTING ECONOMIC AND FINANCIAL TIME SERIES: ARIMA VS. LSTM, SIMA SIAMI NAMIN, https://arxiv.org/pdf/1803.06386.pdf , 2018.
- Deep Learning for Time-Series Analysis, Gamboa J., 2017, https://arxiv.org/pdf/1701.01887.pdf
- G.Peter Zhang, B.Eddy Patuwo, Michael Y. Hu, "A simulation study of artificial neural networks for nonlinear time-series forecasting," Computers & Operations Research, Volume 28, Issue 4, 2001, pp. 381-396, ISSN 0305-0548, https://doi.org/10.1016/S0305-0548(99)00123-9.
- G.Peter Zhang, Min Qi, "Neural network forecasting for seasonal and trend time series," European Journal of Operational Research, Volume 160, Issue 2, 2005, pp. 501-514, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2003.08.037.
- https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/
- Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015
- Bryan Lim and Stefan Zohren, Time Series Forecasting With Deep Learning: A Survey, https://arxiv.org/abs/2004.13408

# References


- MLP
  - Shiblee, M., Kalra, P. K., & Chandra, B. (2009). Time Series Prediction with Multilayer Perceptron (MLP): A New Generalized Error Based Approach. Lecture Notes in Computer Science, 37–44. doi:10.1007/978-3-642-03040-6_5
  - N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados and Yoshua Bengio, https://arxiv.org/pdf/1905.10437.pdf
- RNNs, LSTMs
  - K. A. Althelaya, E. M. El-Alfy and S. Mohammed, "Evaluation of bidirectional LSTM for short-and long-term stock market prediction," 2018 9th International Conference on Information and Communication Systems (ICICS), Irbid, 2018, pp. 151-156, doi: 10.1109/IACS.2018.8355458.
  - DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks, David Salinas, Valentin Flunkert and Jan Gasthaus, https://arxiv.org/pdf/1704.04110.pdf
  - Deep State Space Models for Time Series Forecasting, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, https://papers.nips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdfdeep
- 1D CNN, Causal CNN, Dilated CNN
  - *Oord et al. WaveNet: A Generative Model for Raw Audio. https://arxiv.org/pdf/1609.03499.pdf
  - *S. Bai, J. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. https://arxiv.org/pdf/1803.01271.pdf
  - Chang et al. Dilated Recurrent Neural Networks.https://papers.nips.cc/paper/2017/file/32bb90e8976aab5298d5da10fe66f21d-Paper.pdf
- Hybrid Models (ARIMA-LSTM, LSTM-CNN, Attention-LSTM)
  - Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks, Wuokun et al, 2018, https://dl.acm.org/doi/pdf/10.1145/3209978.3210006
  - A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction, Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang and Garrison Cottrell, https://songdj.github.io/publication/ijcai-17-a/ijcai-17-a.pdf
  - Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting, Rajat Sen, Hsiang-Fu Yu and Inderjit Dhillon, https://papers.nips.cc/paper/2019/file/3a0844cee4fcf57de0c71e9ad3035478-Paper.pdf
  - Hybrid Neural Networks for Learning the Trend in Time Series, Lin et al. 2017, https://infoscience.epfl.ch/record/262447
  - Learning representations from eeg with deep recurrent-convolutional neural networks, Bashivan et al., arXiv preprint arXiv:1511.06448, 2015, https://arxiv.org/pdf/1511.06448.pdf
  - Wang, R., Peng, C., Gao, J. et al. A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series. Comp. Appl. Math. 39, 30 (2020). https://doi.org/10.1007/s40314-019-1006-2
  - G.Peter Zhang, “Time series forecasting using a hybrid ARIMA and neural network model”, Neurocomputing, Volume 50, 2003, pp. 159-175, ISSN 0925-2312, https://doi.org/10.1016/S0925-2312(01)00702-0.
  - Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944
- Transformer for Time-series
  - Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting, Li et al., 2019, https://arxiv.org/pdf/1907.00235.pdf
- Graphs and GNNs
  - Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting, Yaguang Li, Rose Yu, Cyrus Shahabi and Yan Liu, https://arxiv.org/pdf/1707.01926.pdf
  - Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting, Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong and Qi Zhang, https://arxiv.org/pdf/2103.07719.pdf