Advanced Time Series

Lecture 4: **Forecasting III Classification I**

Gleb Ivashkevich

Today

Time series forecasting $\longrightarrow$ classification:

- **N-BEATS model**
- **multi-head attention and transformers**
- **time series classification** problem: setup
- **convolutions**

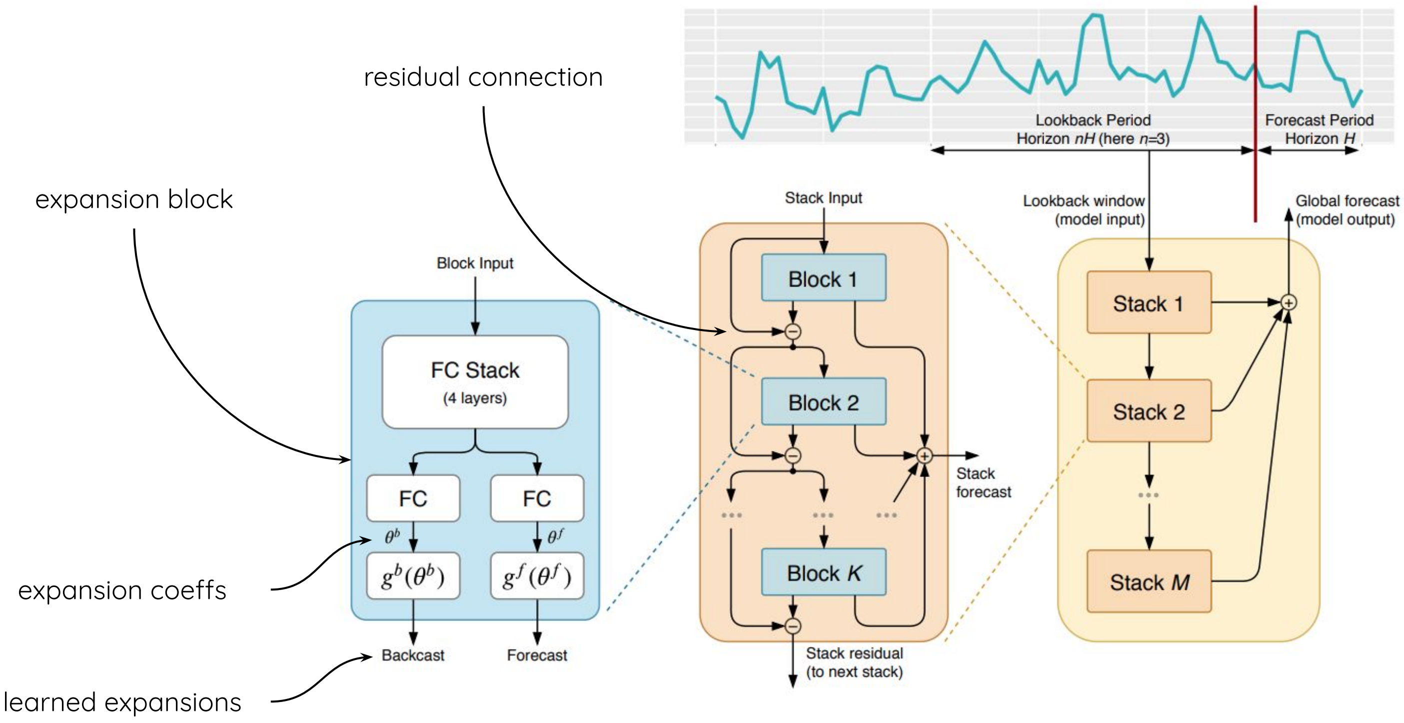
N-BEATS

N-BEATS

[N-BEATS: Neural basis expansion analysis for interpretable time series forecasting]

- an interesting approach, leveraging FC layers, residual connections and stack
- expansion basis can be learned
- no convolutions, no RNN blocks
- not an encoder-decoder

N-BEATS



residual connection
expansion block
expansion coeffs
learned expansions
Block Input
FC Stack
(4 layers)
FC
FC
θ^b
θ^f
$g^b(\theta^b)$
$g^f(\theta^f)$
Backcast
Forecast
Stack Input
Block 1
Block 2
Block K
Stack forecast
Stack residual
(to next stack)
Lookback Period
Horizon nH (here n=3)
Forecast Period
Horizon H
Lookback window
(model input)
Global forecast
(model output)
Stack 1
Stack 2
Stack M

N-BEATS

[N-BEATS: Neural basis expansion analysis for interpretable time series forecasting]

- each block learns the **representation** of it’s input given block’s **“expressivity”**
- next block learns **residual**
- blocks are joined into **stacks**, stacks - into an **ensemble**
- expansion basis can be learned or predefined (seasonality)

Transformer architectures for time series

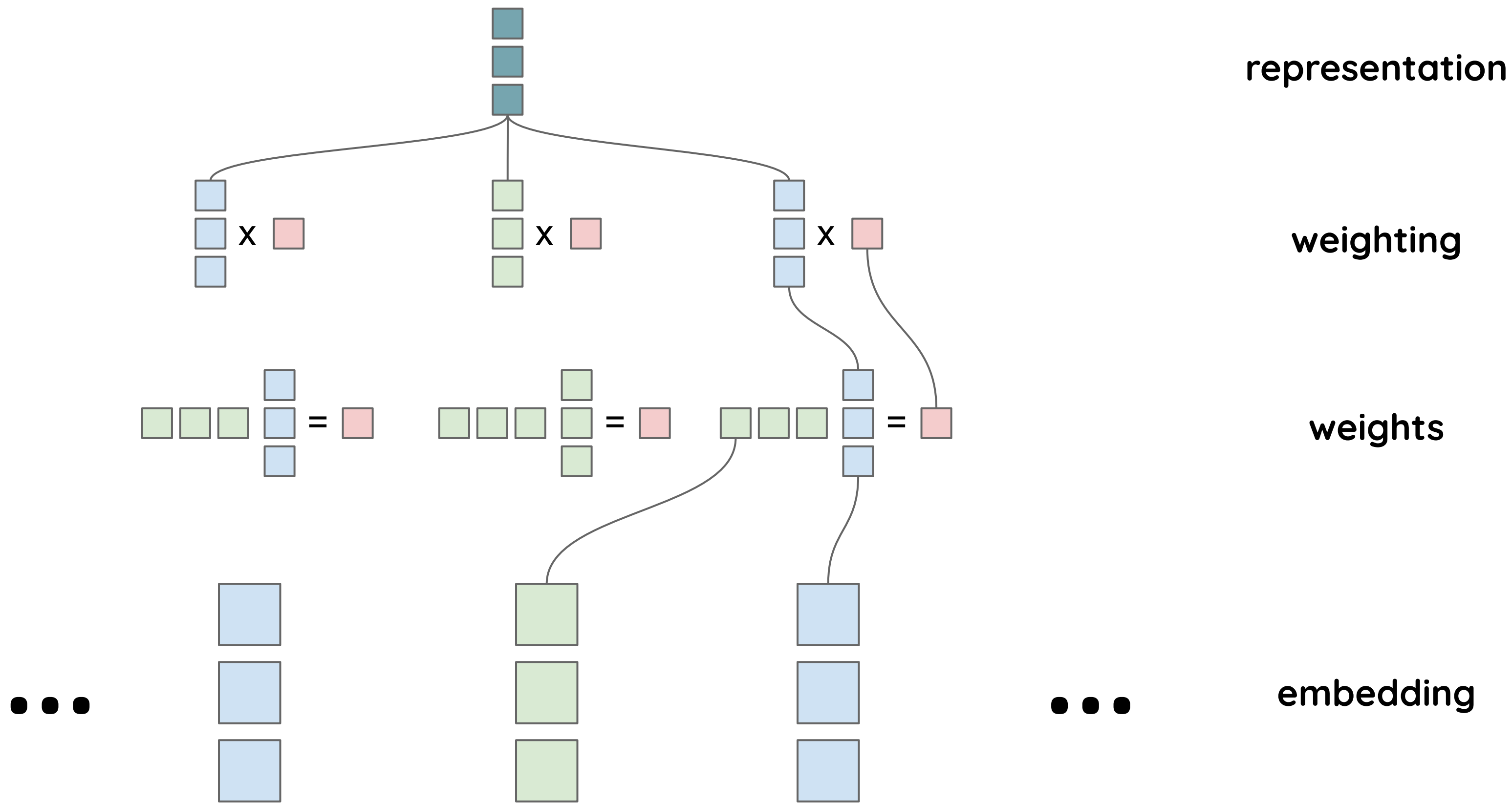
Transformers

Why?

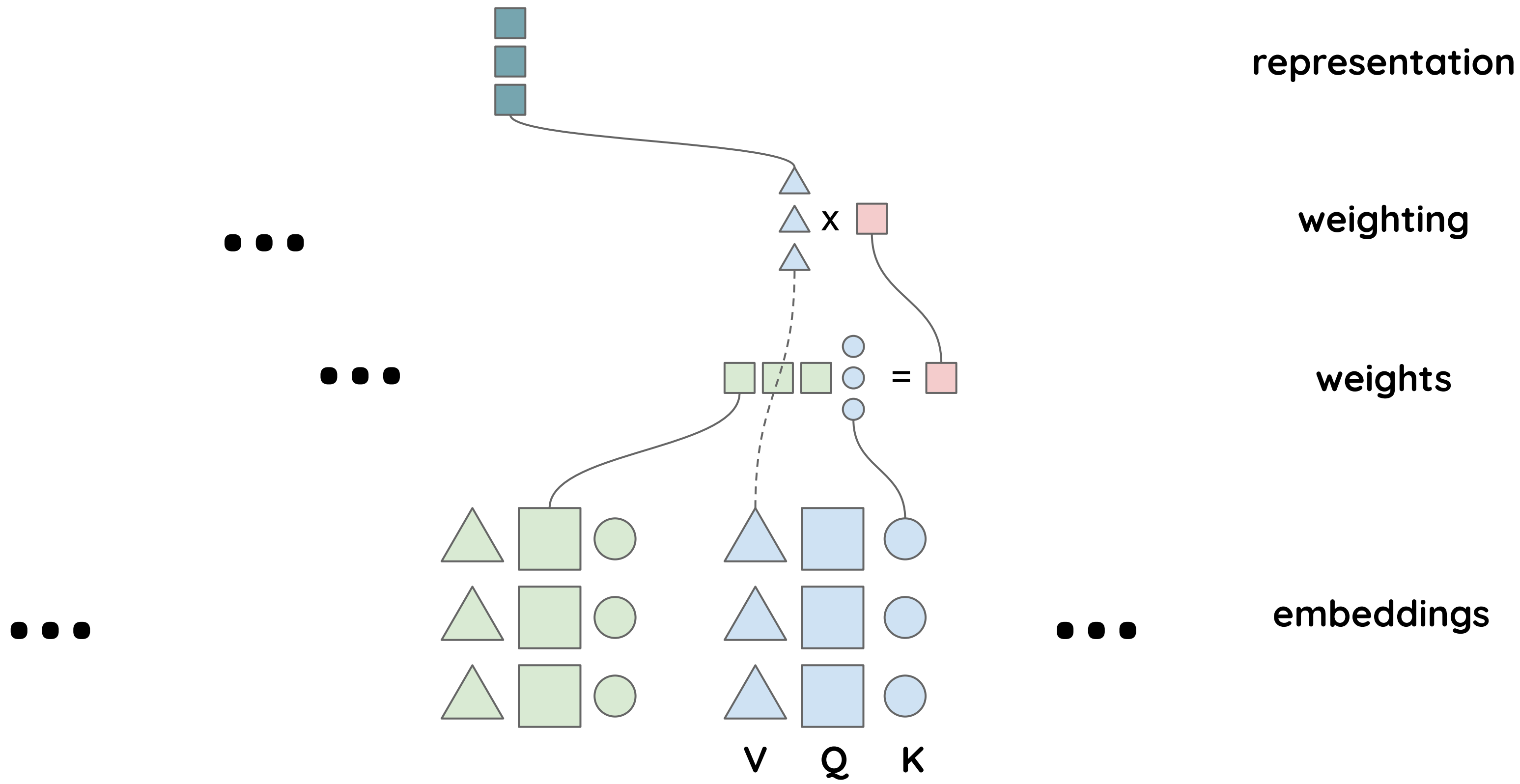
- typical sequential models (RNNs) may still **not catch** temporal dynamics well
- **attention** layer may fix this
- but they are still **sequential**

Transformers are still **encoder-decoder**.

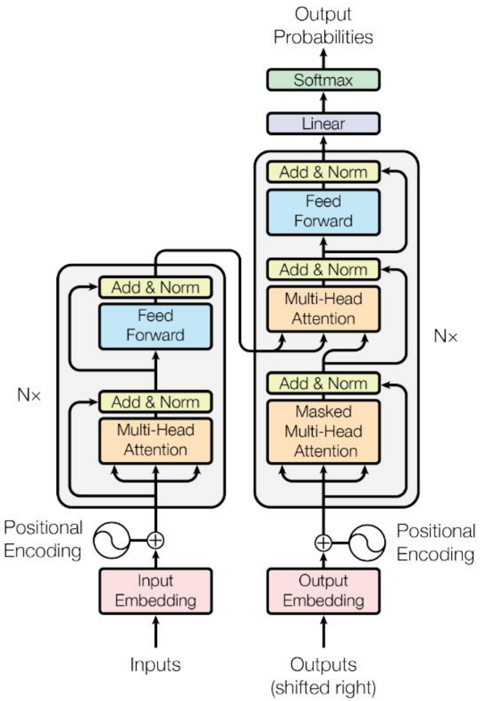
Attention



Multi-head attention



Positional encoding



- allows to attend both absolute and relative positions
- additive!

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Image: [Attention Is All You Need]

Transformers: forecasting

[Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting]

- innovation: **convolutional attention**
  (queries, keys and values are computed by conv layer)

- very good performance compared to other architectures

Transformers: forecasting

[Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting]

- innovation: **convolutional attention**
  (queries, keys and values are computed by conv layer)

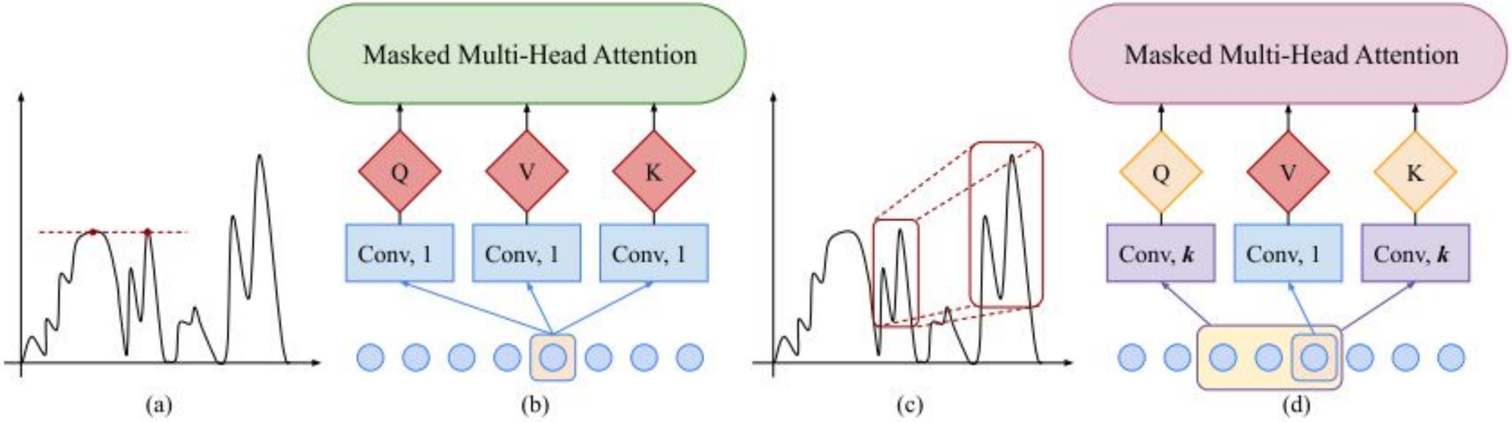
- **attention memory bottleneck:** use smart masking
- **learnable positional encodings**
- **simpler attention**

Transformers: forecasting

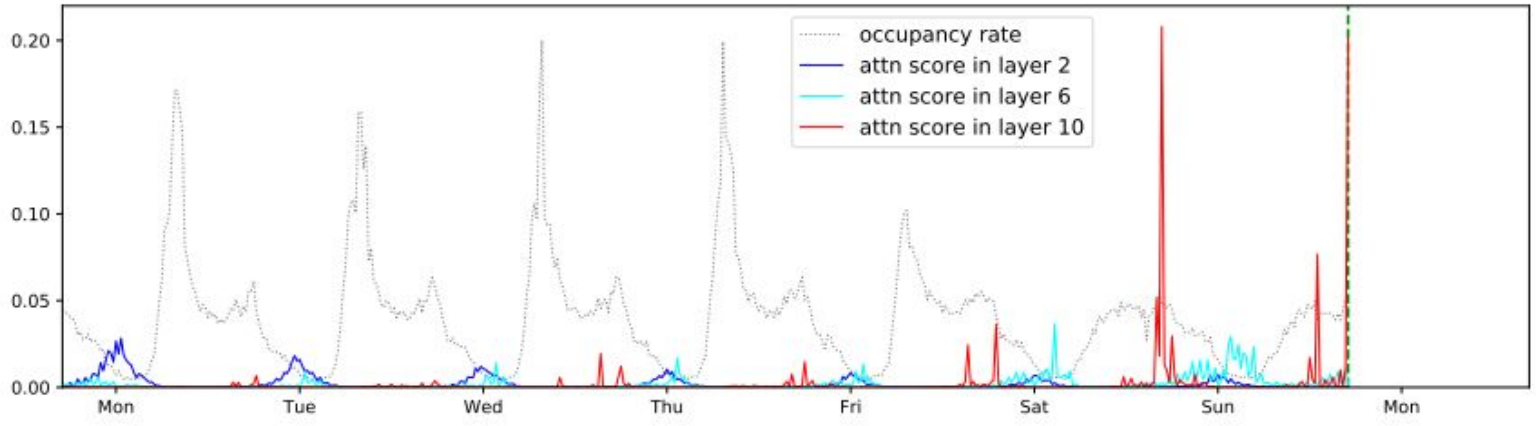
[Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting]

- decoder-only mode: **similar to DeepAR**
- **probabilistic forecasts** (DeepAR inspired)
- **causal convolutions**
- hourly power consumption dataset

Transformers: forecasting

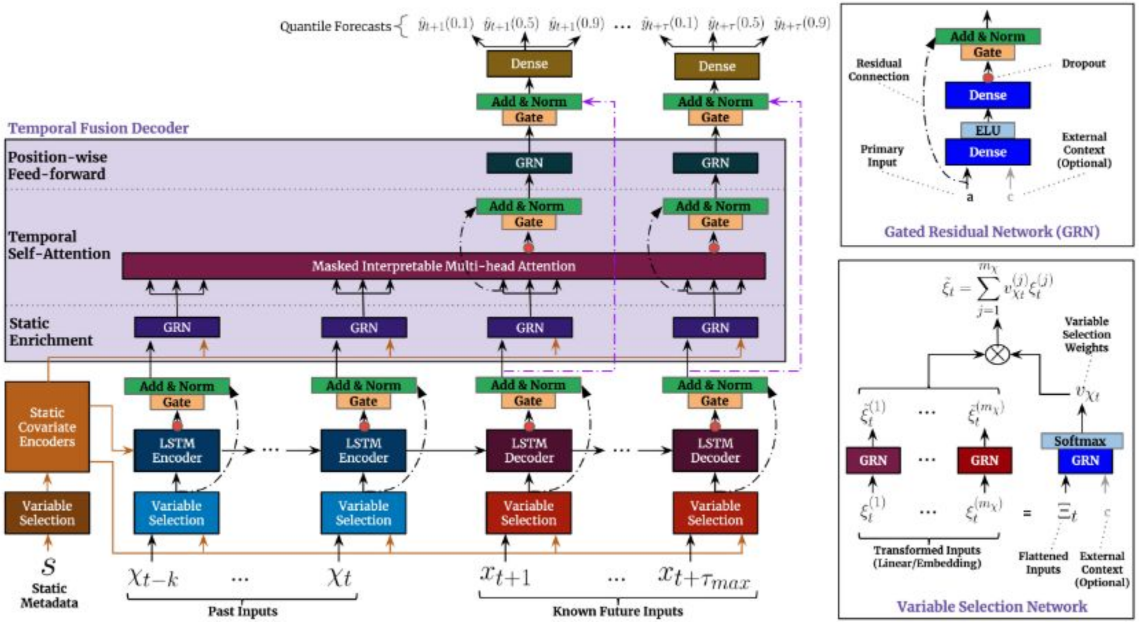


Transformers: forecasting



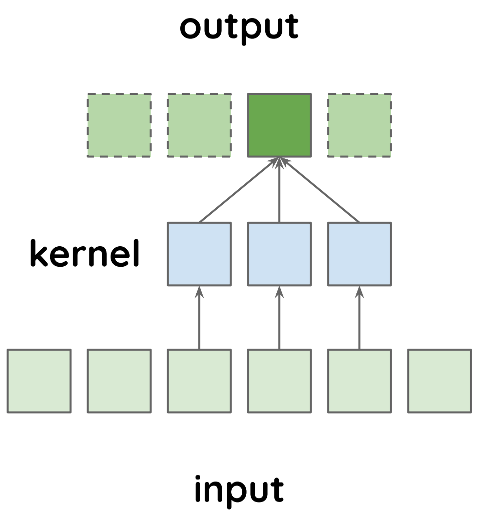
Other papers

[Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting]

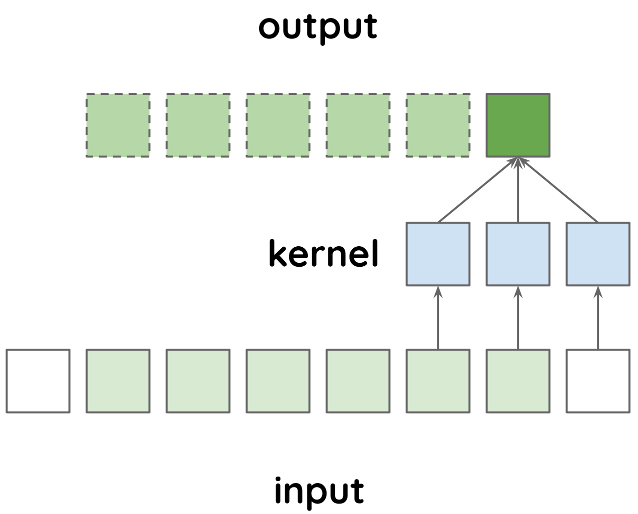


Dilated and causal convolutions

Convolution

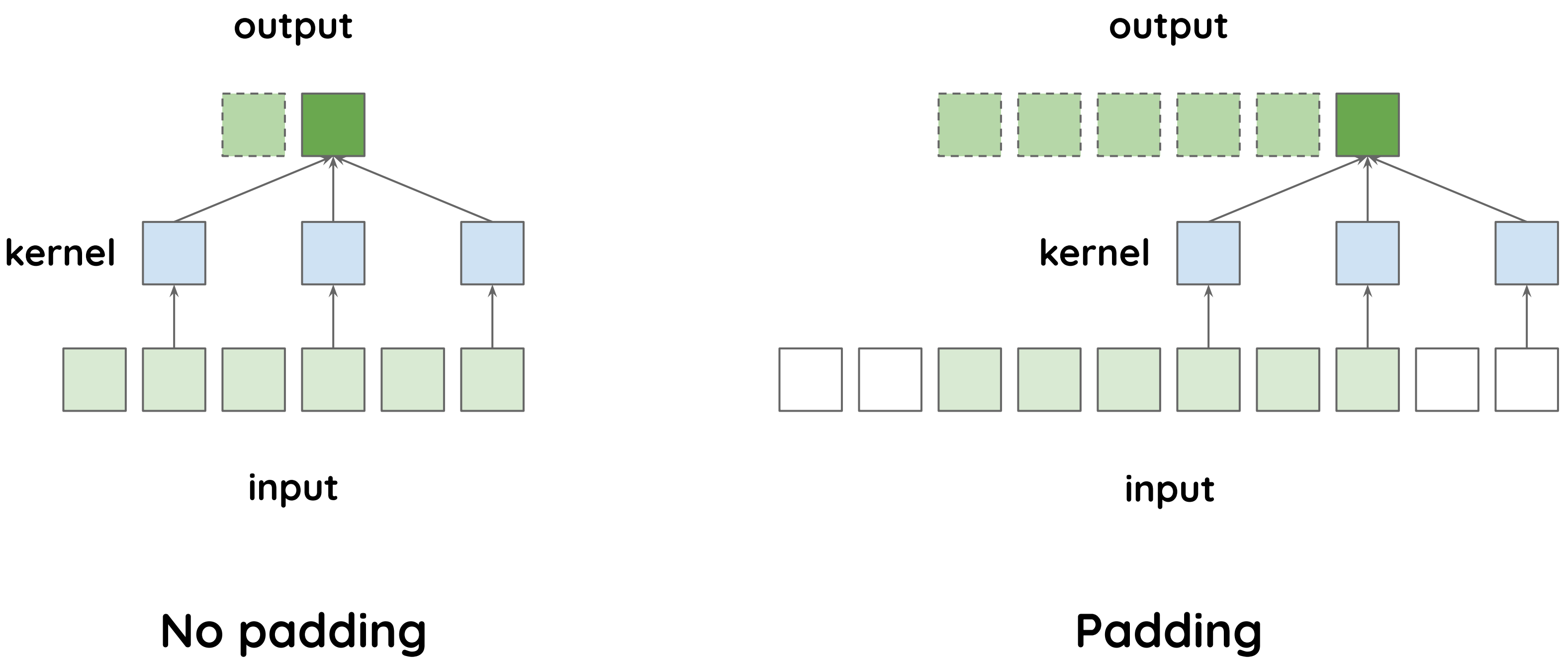


No padding



Padding

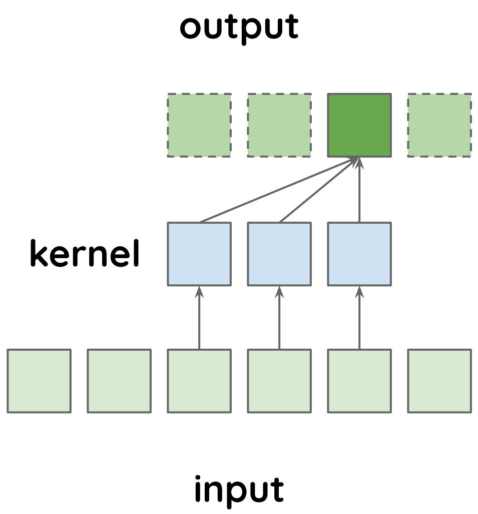
Dilated convolution



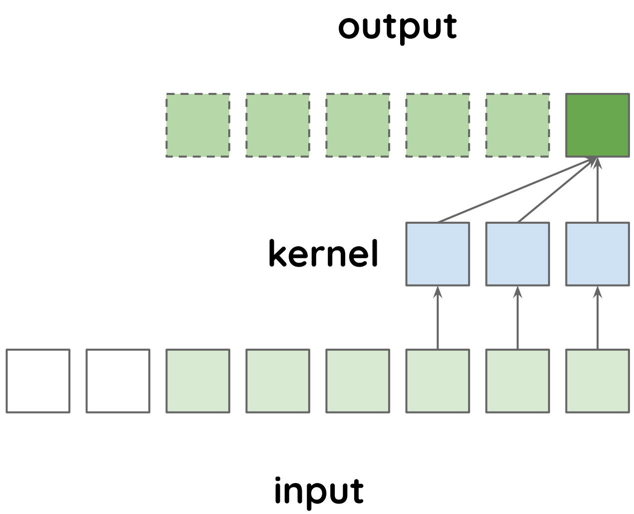
Why dilated convolutions

- fast extension of receptive field
- no additional computational costs
- high resolution input is manageable (high-frequency t. s.)

Causal convolution

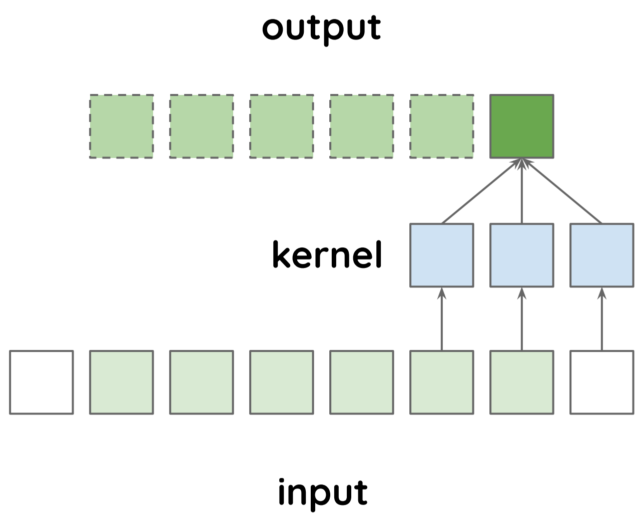


No padding

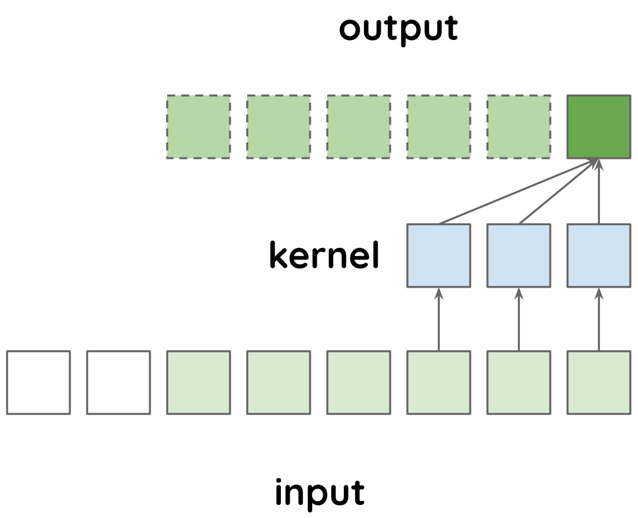


Padding

Causal convolutions impl.

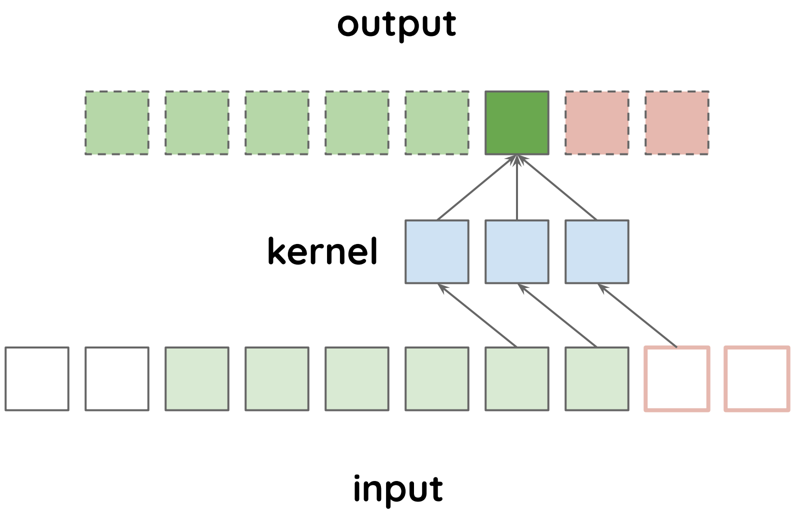


non-causal

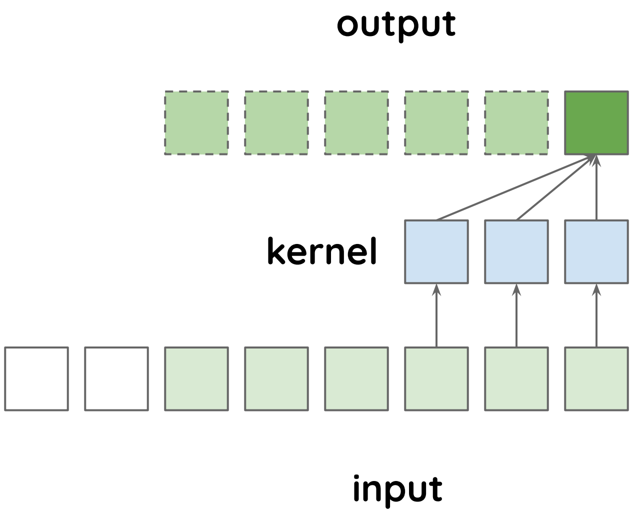


causal:
not available in PyTorch

Causal convolutions impl.



non-causal,
overpadded



causal

Transformers

Benefits:

- allow for parallelization
- do not limit other architectural ideas
- add interpretability proxy
- much easier to reason about

Time series classification

Classification

Setup:

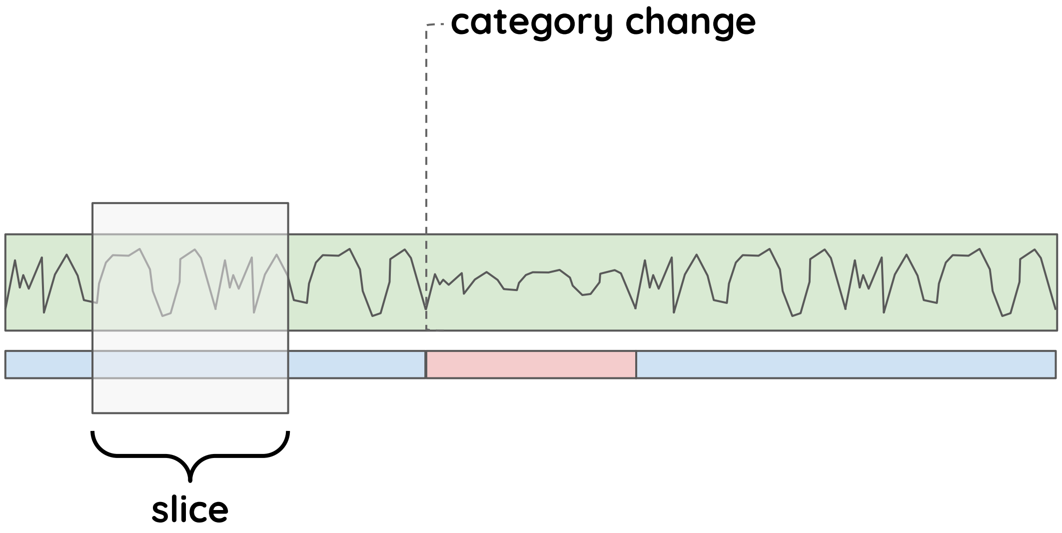
- given a **slice** of (usually multivariate) time series, get it's **category**
- **wide variety** of slice duration and typical time scales
- **patterns**, not long-term dependencies
- hence, convolutions
- generally, **conceptually simpler** compared to forecasting

Patterns

Setup:

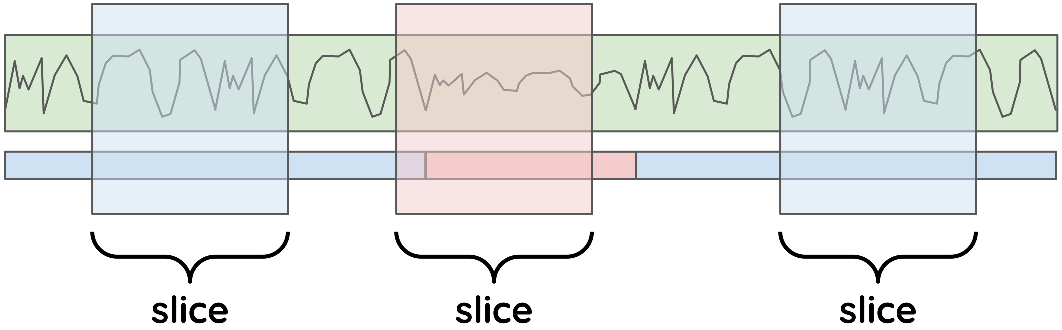
- given a **slice** of (usually multivariate) time series, get it's **category**
- **wide variety** of slice duration and typical time scales
- **patterns**, not long-term dependencies
- hence, convolutions

Patterns

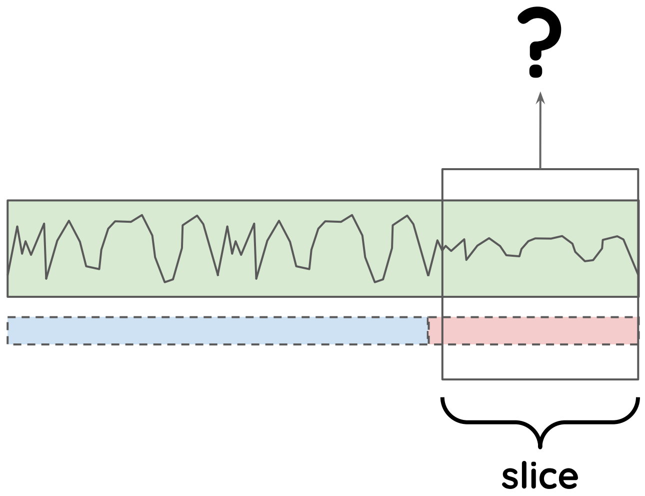


There are **no categories** if changes are **too gradual** and can be modeled with recurrent networks

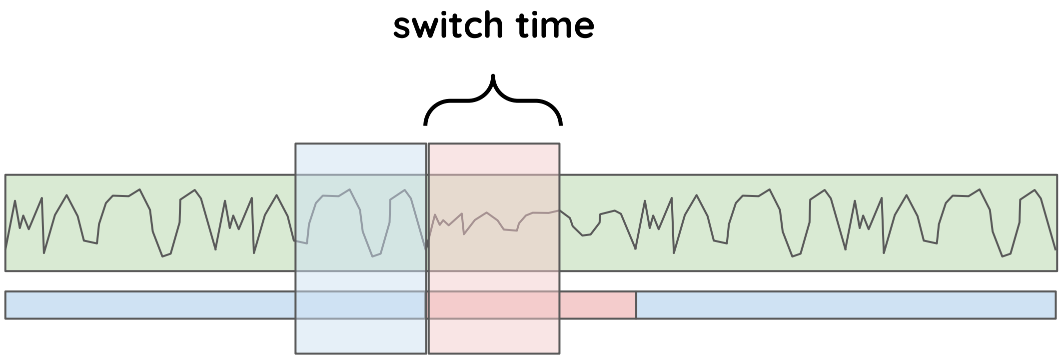
Training setup



Inference setup



Switch time



Switch time is about the size of the **classification window**. Must be **aligned** carefully to category duration time.

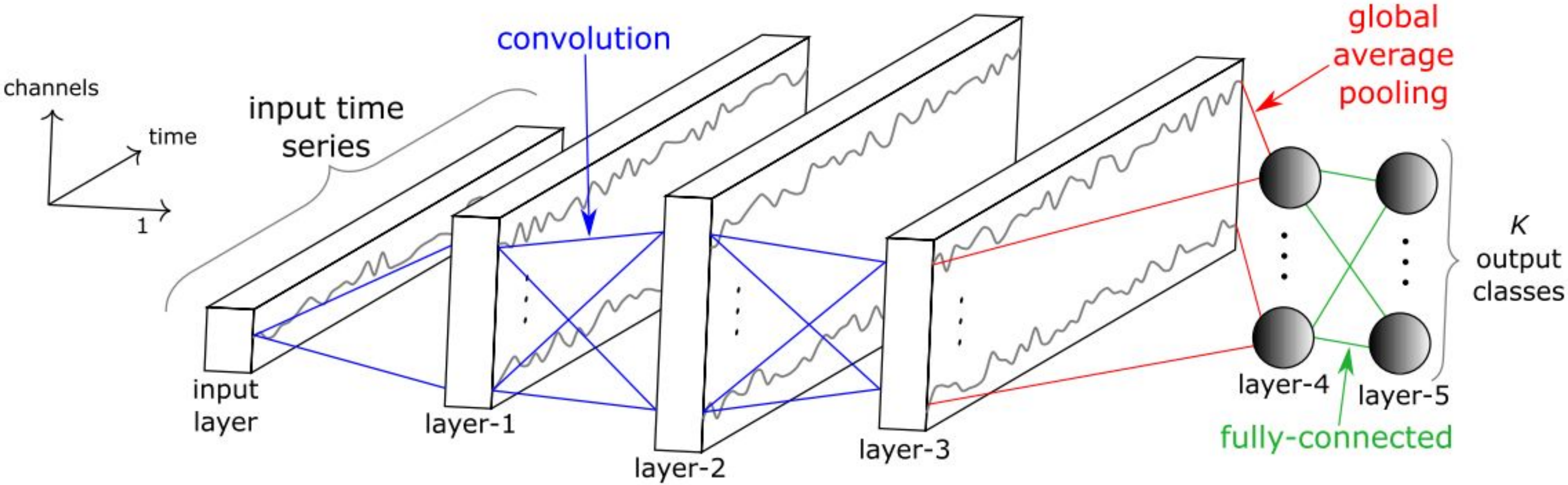
Classical

A lot of approaches:

- manually created windowed features + classical models
- **DTW** (dynamic time warping) as a distance measure
- etc.

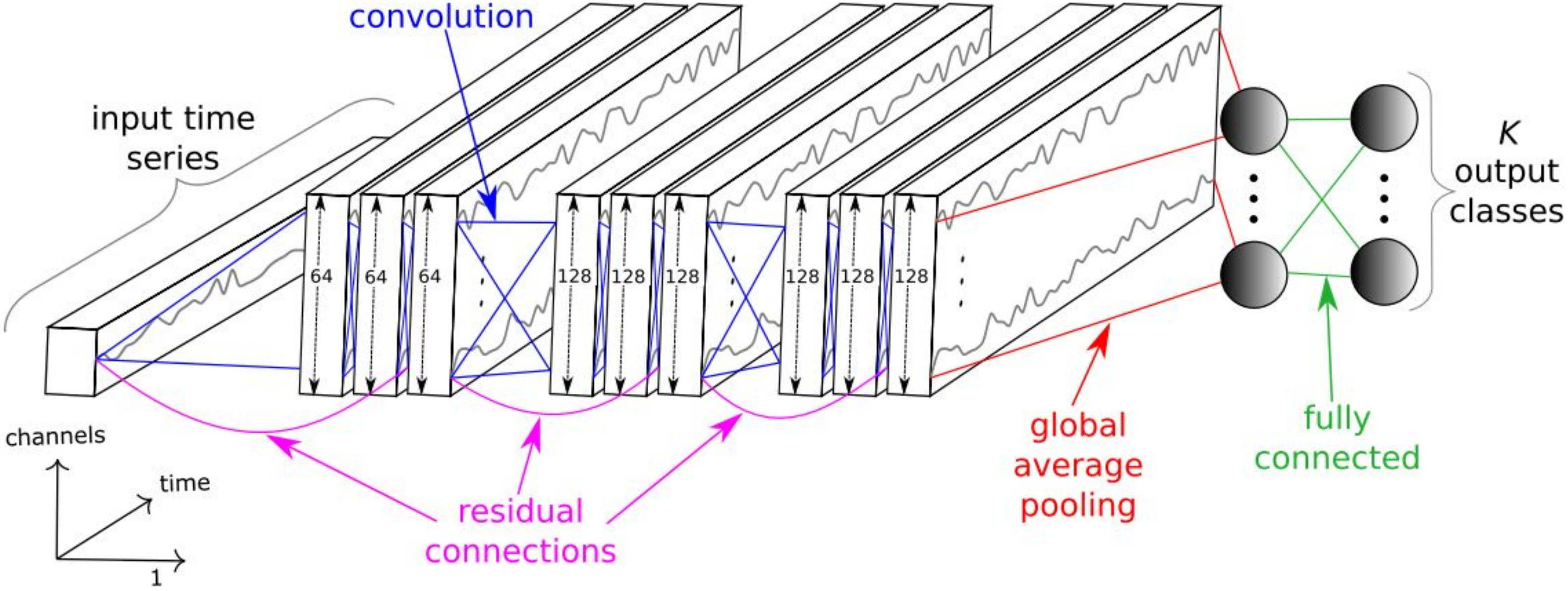
Classification architectures

Fully convolutional



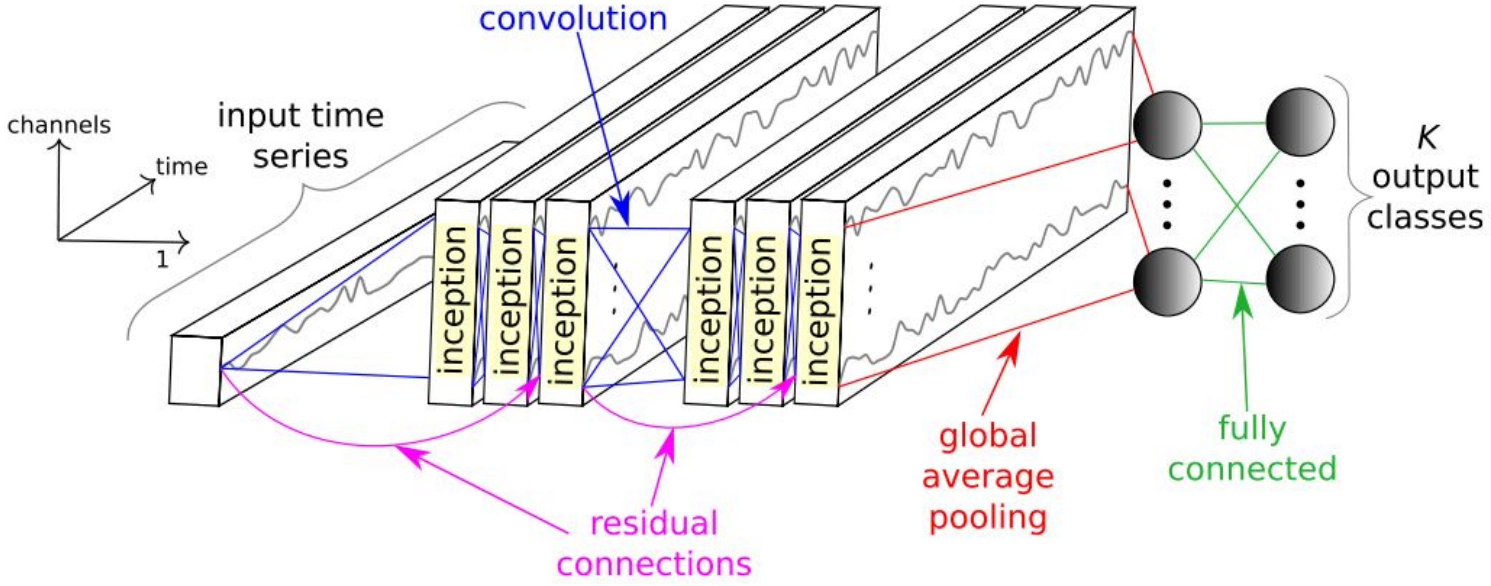
- Pictures: [Deep learning for time series classification: a review]

Residual



- Pictures: [Deep learning for time series classification: a review]()

InceptionTime



- Picture: [InceptionTime: Finding AlexNet for Time Series Classification]

Next time

- InceptionTime implementation
- classification for t. s. segmentation

questions?