Time Series Forecasting

3.3 Convolutional Networks

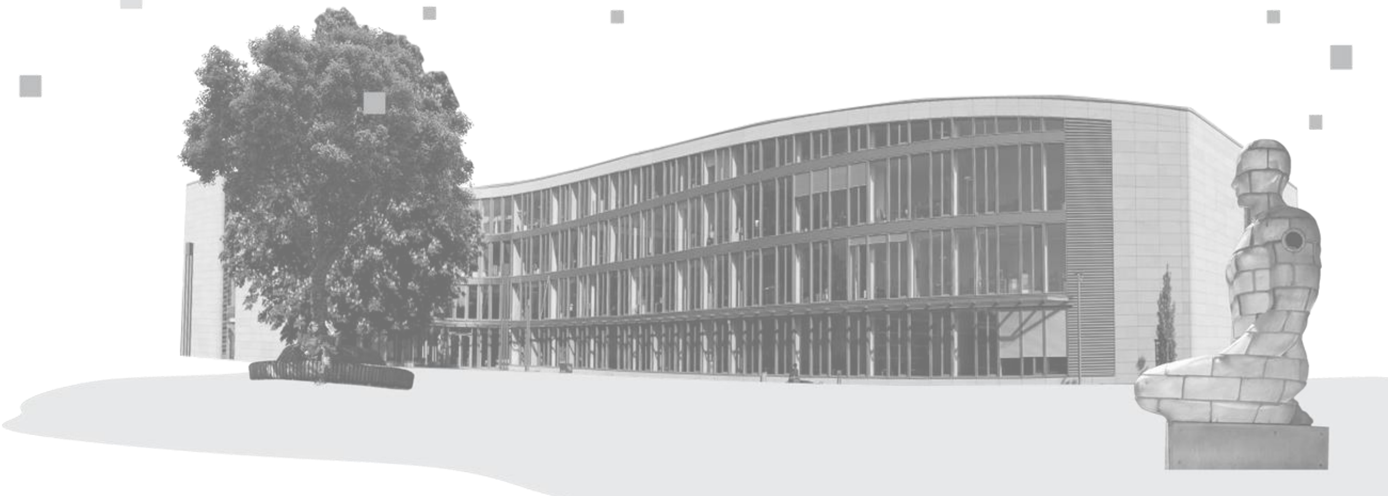
Mario Tormo Romero

KI Service Zentrum by Hasso-Plattner-Institut

HPI

Gefördert durch:

Bundesministerium für Forschung, Technologie und Raumfahrt



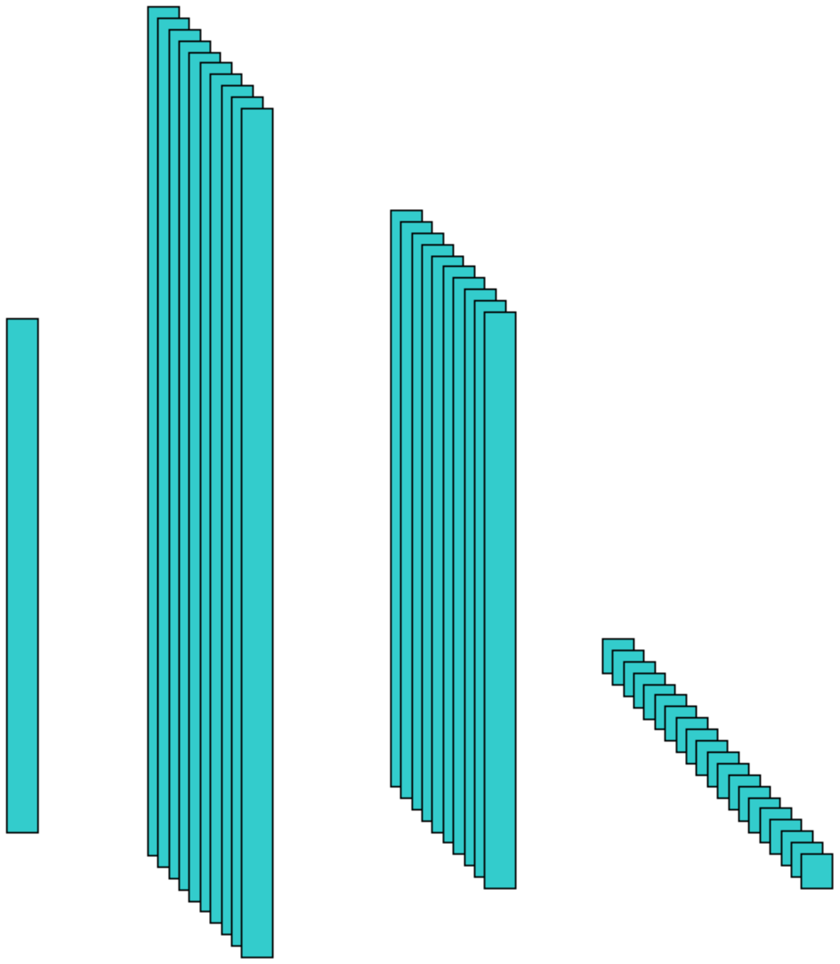
Design IT.
Create Knowledge.

What we`ll cover in this video

- Why CNNs are useful for time series data
- Key concepts:
  - 1D and dilated convolutions
  - Residual connections and pooling
- Introduction to Temporal Convolutional Networks (TCNs)
- Practical example: Building a CNN model for forecasting

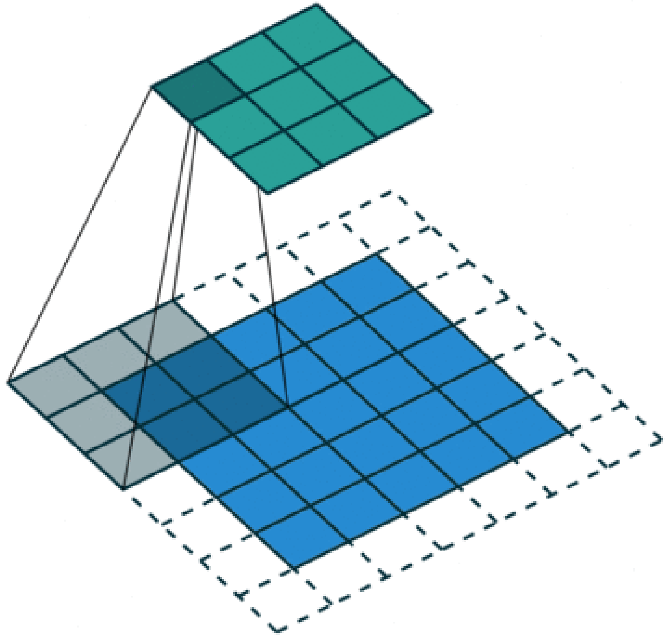
Why CNNs Are Useful for Time Series Forecasting

- Pattern detection through sliding filters
- Parallel processing of sequences — faster training compared to RNNs
- Reduced vanishing/exploding gradient issues in deeper models
- Efficient capture of local temporal patterns
- Dilated convolutions expand receptive field for long-range dependencies



What Are Convolutional Neural Networks?

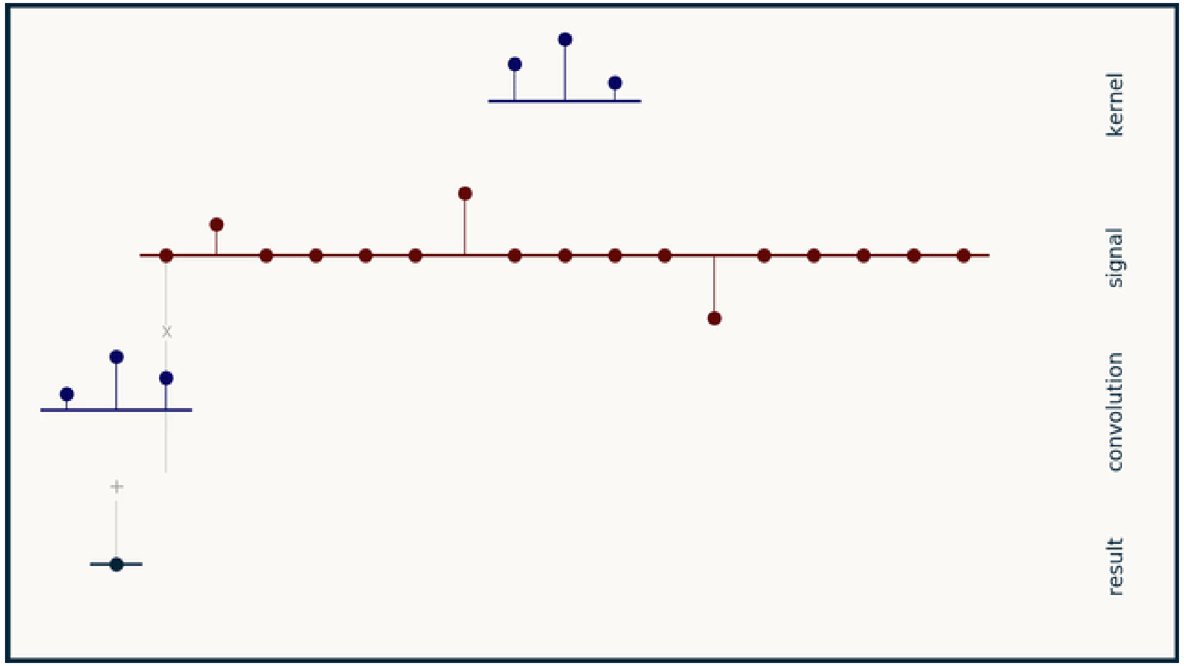
- Designed to automatically and adaptively learn spatial and temporal patterns through convolutional layers.
- How They Work:
  - Apply filters (kernels) that slide over input data to extract local features.
  - Each filter detects specific patterns (like edges in images or temporal trends in sequences).
  - Multiple filters produce different feature maps, capturing diverse characteristics.
  - Stacking convolutional layers enables learning of increasingly complex patterns.



- Key Characteristics:
- **Local Connectivity**: Focus on small neighborhoods in data at a time.
- **Parameter Sharing**: Same filter used across the entire input reduces the number of parameters.
- **Hierarchical Feature Learning**: Lower layers learn simple features, higher layers learn complex abstractions.

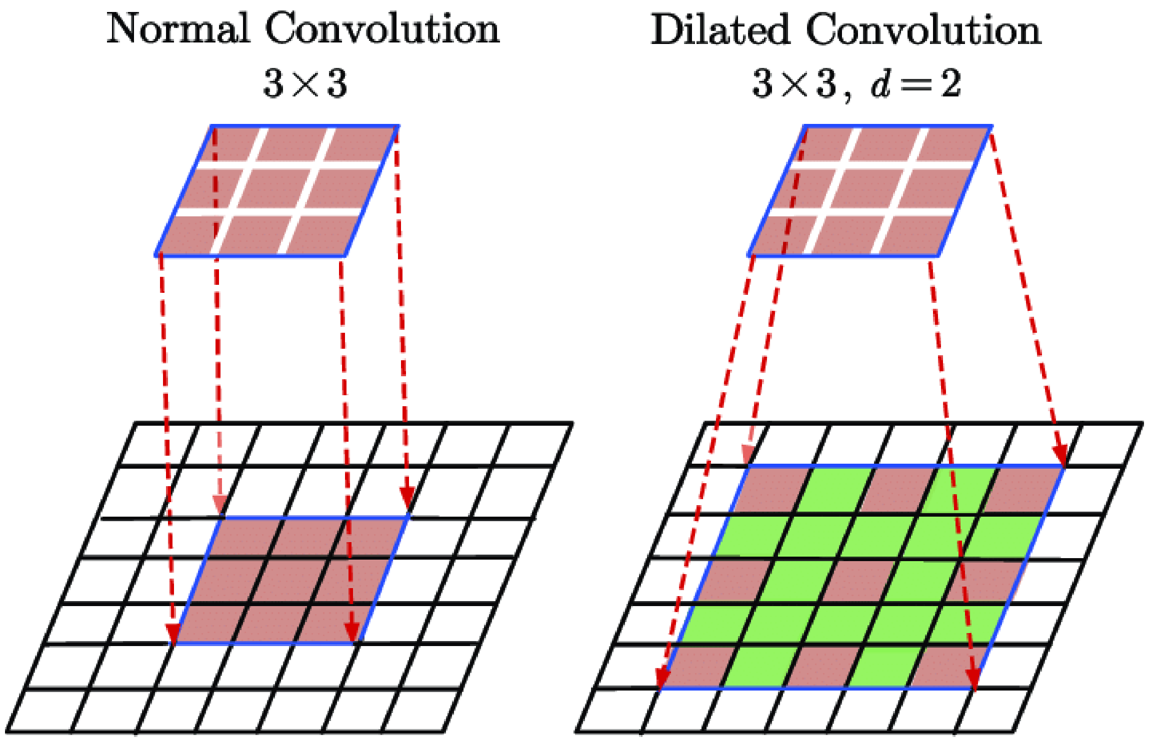
Core Components of CNNs for Time Series Forecasting

- **1D Convolutions**
  - Filters slide along the time axis
  - Extract local temporal patterns
- **Dilated Convolutions**
  - Skip input points to expand receptive field
  - Capture longer-range dependencies without deeper networks
- **Residual Connections**
  - Bypass pathways to help train deeper CNNs
  - Prevent vanishing gradients and information loss
- **Pooling Layers**
  - Downsample the feature maps
  - Reduce noise and computational cost



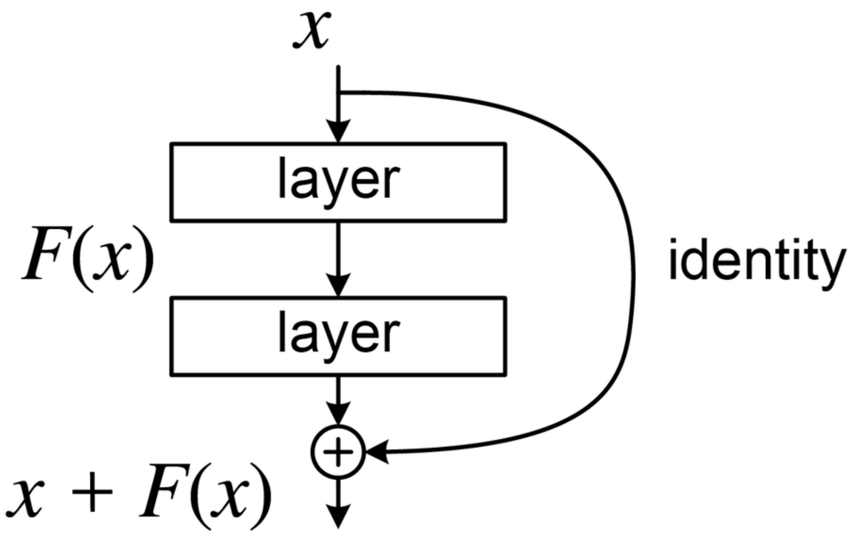
Core Components of CNNs for Time Series Forecasting

- **1D Convolutions**
  - Filters slide along the time axis
  - Extract local temporal patterns
- **Dilated Convolutions**
  - Skip input points to expand receptive field
  - Capture longer-range dependencies without deeper networks
- **Residual Connections**
  - Bypass pathways to help train deeper CNNs
  - Prevent vanishing gradients and information loss
- **Pooling Layers**
  - Downsample the feature maps
  - Reduce noise and computational cost



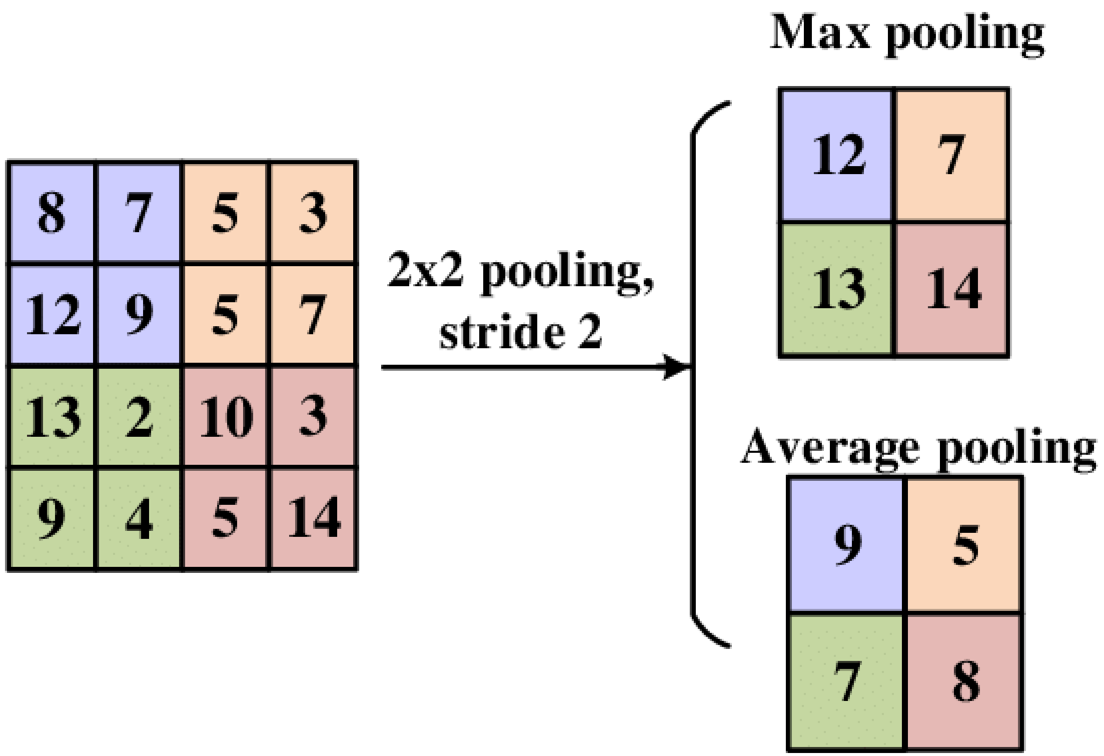
Core Components of CNNs for Time Series Forecasting

- **1D Convolutions**
  - Filters slide along the time axis
  - Extract local temporal patterns
- **Dilated Convolutions**
  - Skip input points to expand receptive field
  - Capture longer-range dependencies without deeper networks
- **Residual Connections**
  - Bypass pathways to help train deeper CNNs
  - Prevent vanishing gradients and information loss
- **Pooling Layers**
  - Downsample the feature maps
  - Reduce noise and computational cost



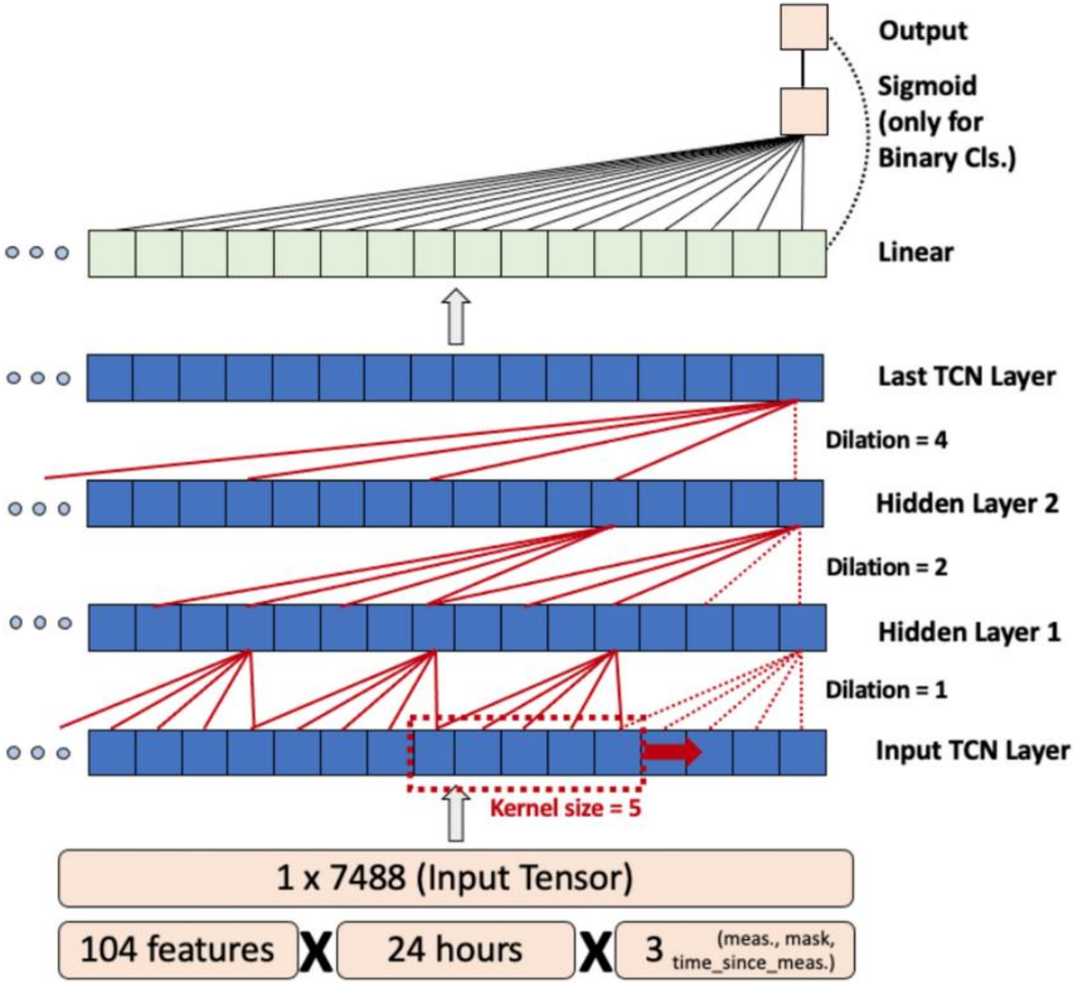
Core Components of CNNs for Time Series Forecasting

- **1D Convolutions**
  - Filters slide along the time axis
  - Extract local temporal patterns
- **Dilated Convolutions**
  - Skip input points to expand receptive field
  - Capture longer-range dependencies without deeper networks
- **Residual Connections**
  - Bypass pathways to help train deeper CNNs
  - Prevent vanishing gradients and information loss
- **Pooling Layers**
  - Downsample the feature maps
  - Reduce noise and computational cost



Temporal Convolutional Networks (TCNs)

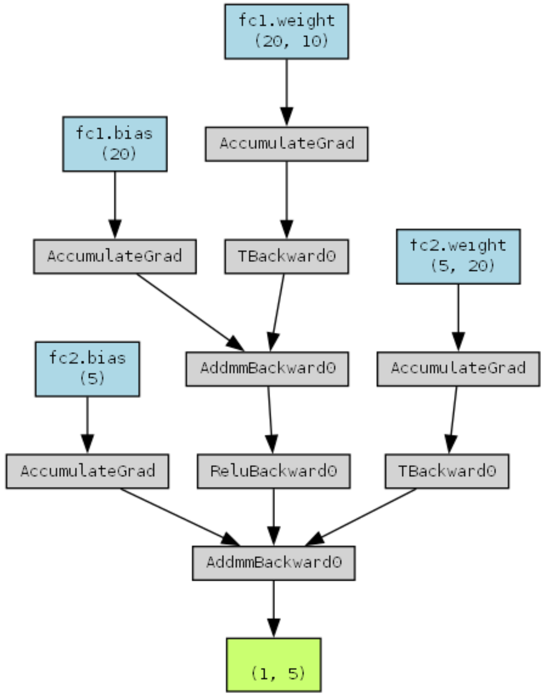
- CNN architecture specifically designed for sequence modeling
- Uses **causal convolutions** to ensure no future information leaks into predictions
- Employs **dilated convolutions** to capture long-range dependencies
- Incorporates **residual connections** for stable and deep networks
- **Parallelizable**: faster training than RNNs/LSTMs
- Avoid vanishing gradient problems common in recurrent models
- Often achieve state-of-the-art performance on sequence tasks



Practical Example — Implementing a CNN for Time Series Forecasting

- Dataset: Univariate time series from **Berlin Daily Temperature Dataset**
- Model Architecture:
  - **Input Layer** (samples, time steps, features)
  - **1D Convolutional Layer** (extract local temporal patterns)
  - **Max Pooling Layer** (reduce dimensionality)
  - **Fully Connected Layer** (map to forecast output)
- Framework: **Python** + **PyTorch**
- **Goal**: Predict the next value from past observations

```
CNNForecast(
  (conv1): Conv1d(1, 16, kernel_size=(3,), stride=(1,))
  (relu): ReLU()
  (pool): MaxPool1d(kernel_size=2, stride=2, padding=0,
                    dilation=1, ceil_mode=False)
  (fc): Linear(in_features=224, out_features=1, bias=True)
)
```



What we‘ve learnt

- CNNs efficiently capture local temporal patterns in time series data.
- 1D and dilated convolutions expand receptive fields without extra parameters.
- Temporal Convolutional Networks (TCNs) use causal convolutions and residual blocks for better long-range dependency modeling.
- CNNs enable faster, parallel training compared to RNNs.
- Practical CNN models for forecasting include convolutional, pooling, and dense layers implemented in frameworks like PyTorch.