4TU.

AI-Enabled Forecasting for Energy System Control

Elham (Eli) Shirazi, e.shirazi@utwente.nl
Assistant Professor, University of Twente

TU Delft Delft University of Technology
TU/e Eindhoven University of Technology
University of Twente.
Wageningen University & Research
4TU+
university of groningen



Forecasting

Reading through the crystal ball …



Forecasting in Energy systems

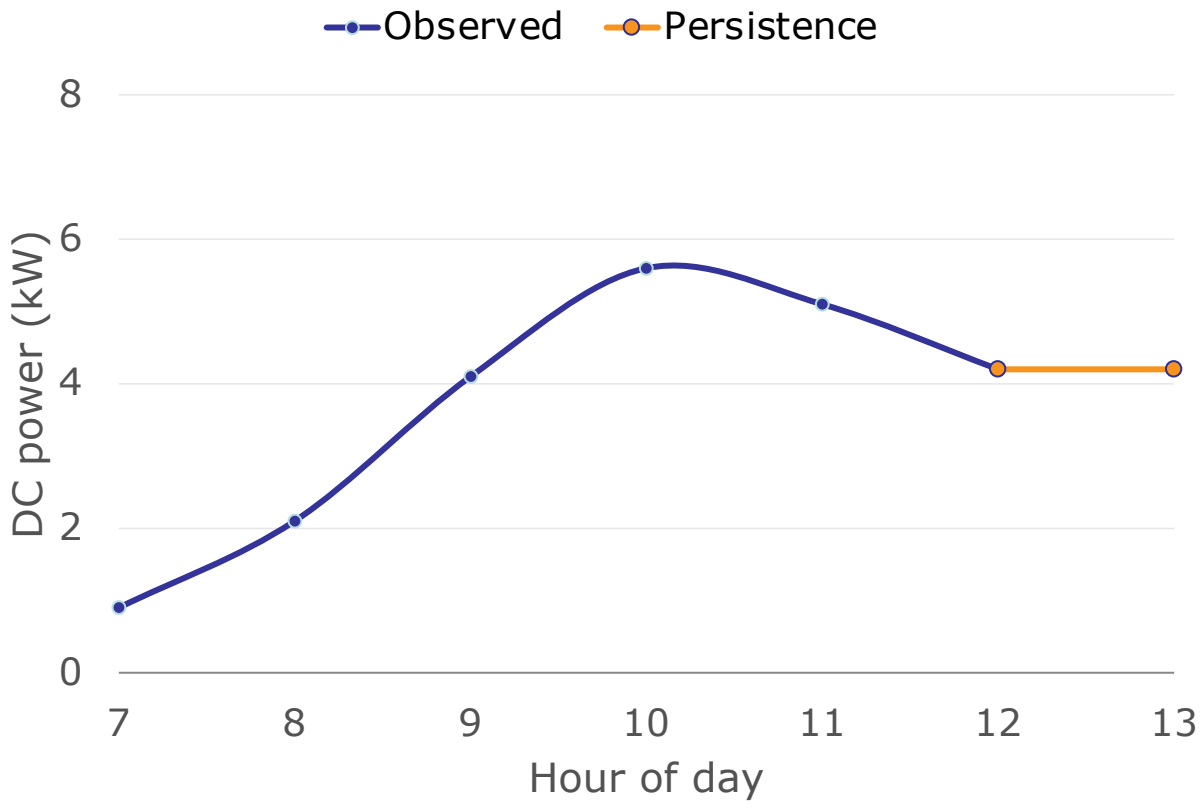


Time-series forecasting

- y(t): observed value at time t
- ŷ(t): forecasted value at time t

Persistence forecast

$\hat{y}(t+1) = y(t)$
Persistence or naïve



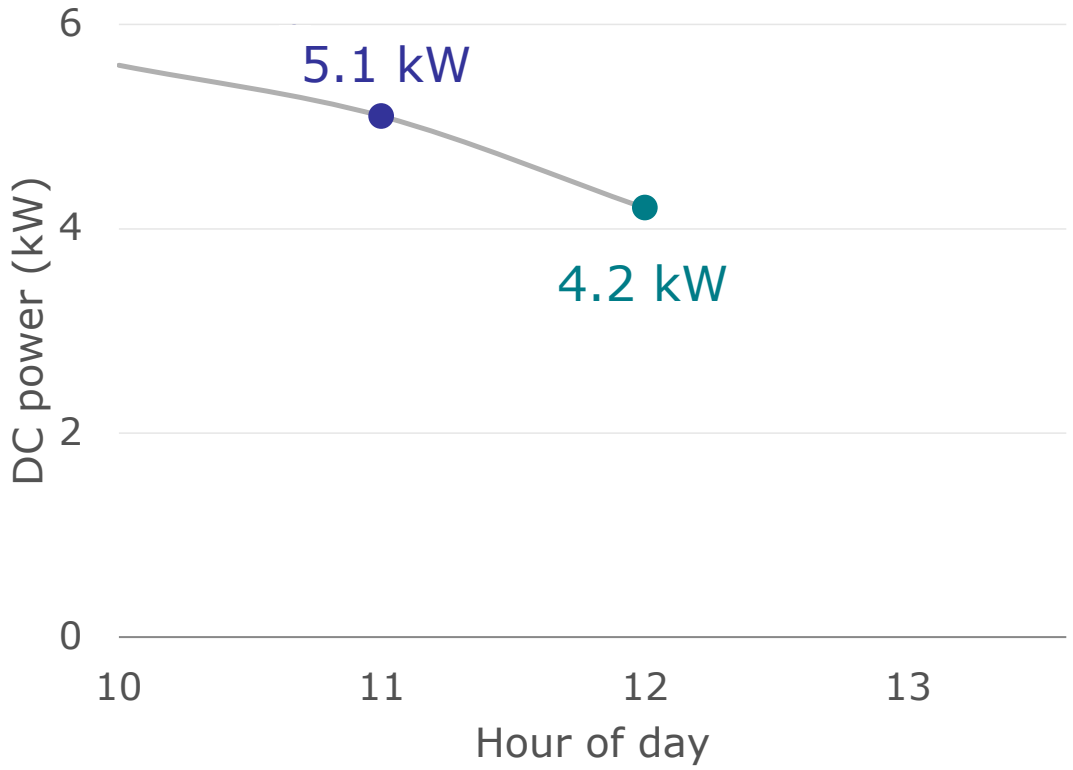
4TU.

STATISTICAL MODELS

Autoregression (AR)

$$\hat{y}(t+1) = c + \varphi_1 y(t) + \varphi_2 y(t-1)$$

$c = 0.5$, $\varphi_1 = 0.6$, $\varphi_2 = 0.3$

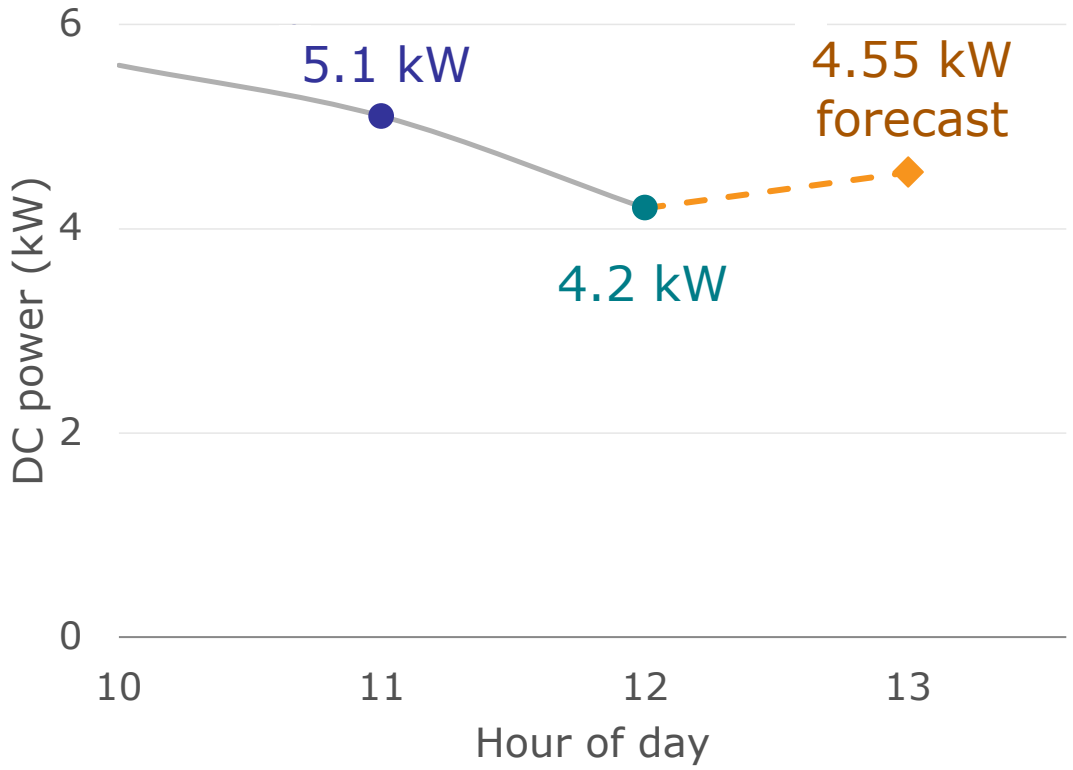


Autoregression (AR)

$\hat{y}(t+1) = c + \varphi_1 y(t) + \varphi_2 y(t-1)$

$c = 0.5, \varphi_1 = 0.6, \varphi_2 = 0.3$

$\hat{y}(13{:}00) = 0.50 + 0.60 \times 4.2 + 0.30 \times 5.1 = 4.55$ kW



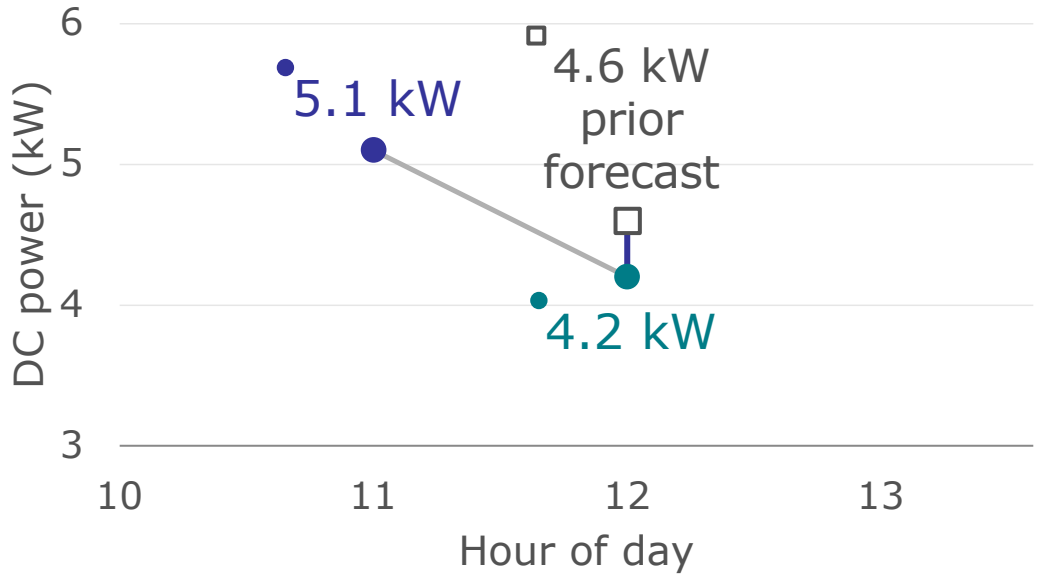
Autoregressive moving average (ARMA)

$\hat{y}(t+1) = c + \varphi\, y(t) + \theta\, \hat{e}(t)$

$\theta = 0.50$, $\varphi = 0.90$, $c = 0.5$

(MA of prediction error)
Latest prediction error

$\hat{e}(t) = 4.2 - 4.6 = -0.40$ kW



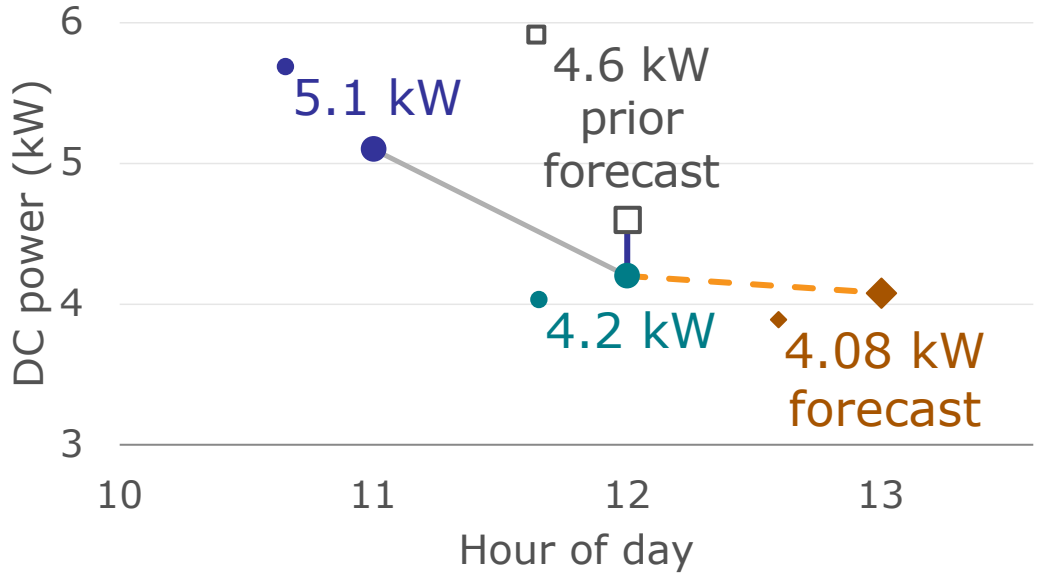
Autoregressive moving average (ARMA)

$\hat{y}(t+1) = c + \varphi\, y(t) + \theta\, \hat{e}(t)$

$\theta = 0.50$, $\varphi = 0.90$, $c = 0.5$

(MA of prediction error)
Latest prediction error

$\hat{e}(t) = 4.2 - 4.6 = -0.40$ kW



$\hat{y}(13{:}00) = 0.50 + 0.90 \times 4.2 + 0.5 \times (-0.40) = 4.08$ kW

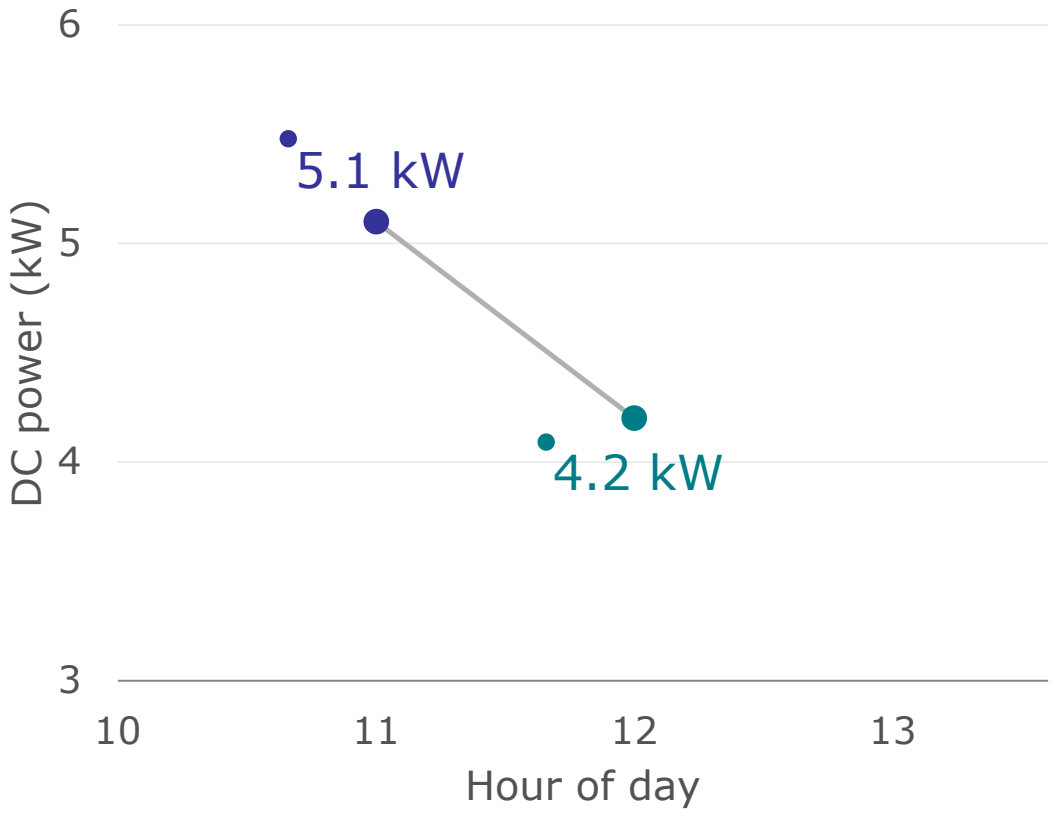
Autoregressive integrated moving average (ARIMA)

$$\Delta y(t) = y(t) - y(t-1)$$
$$\Delta \hat{y}(t+1) = \varphi \, \Delta y(t)$$
$$\hat{y}(t+1) = y(t) + \Delta \hat{y}(t+1)$$

$\varphi = 0.5$

$\Delta y(t) = 4.2 - 5.1 = -0.90$

$\Delta \hat{y}(t+1) = 0.50 \times (-0.90) = -0.45$



Autoregressive integrated moving average (ARIMA)

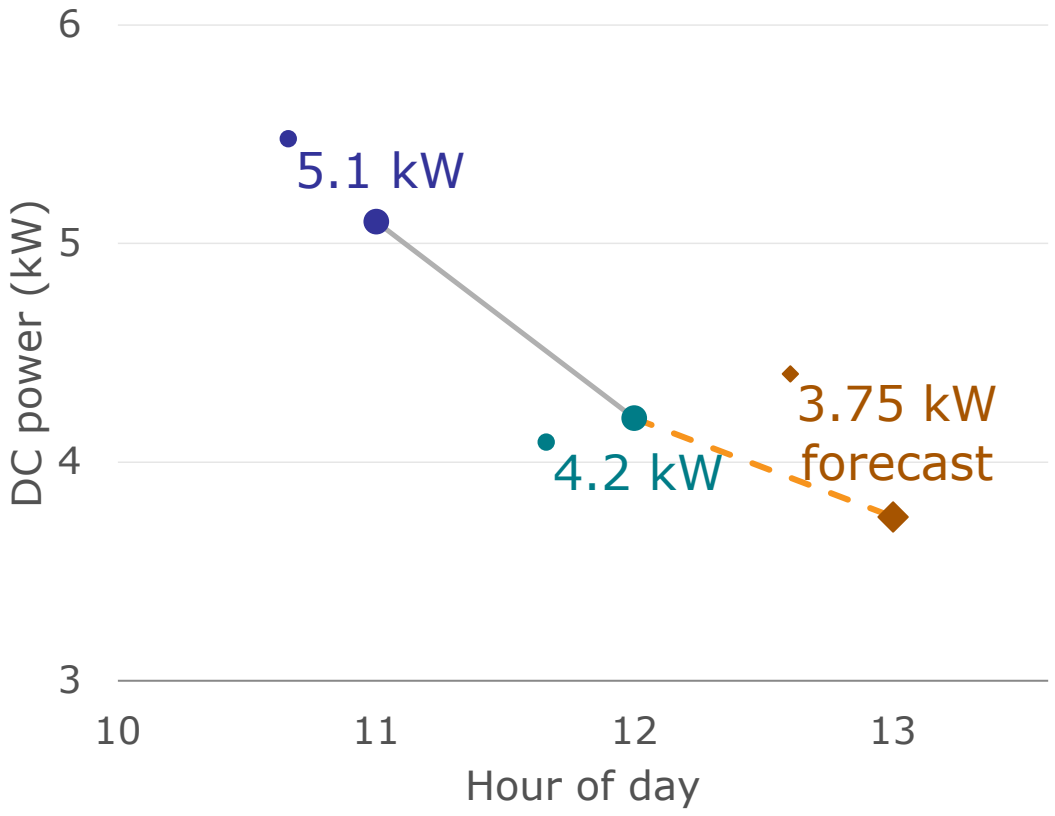
$$\Delta y(t) = y(t) - y(t-1)$$
$$\Delta \hat{y}(t+1) = \varphi \, \Delta y(t)$$
$$\hat{y}(t+1) = y(t) + \Delta \hat{y}(t+1)$$

$\varphi = 0.5$

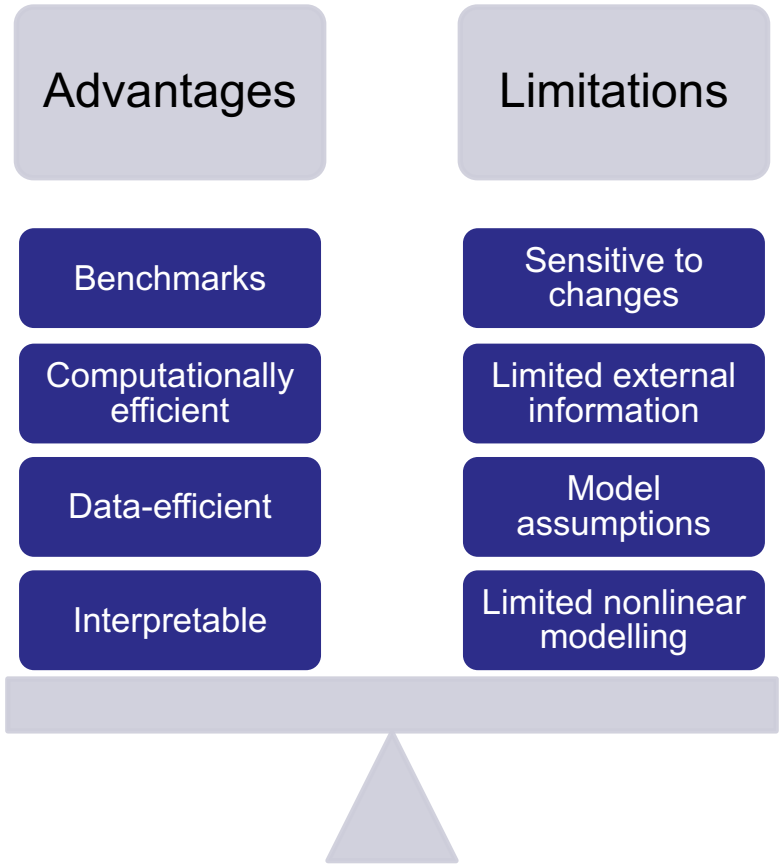
$\Delta y(t) = 4.2 - 5.1 = -0.90$

$\Delta \hat{y}(t+1) = 0.50 \times (-0.90) = -0.45$

$\hat{y}(13{:}00) = 4.2 - 0.45 = 3.75$ kW



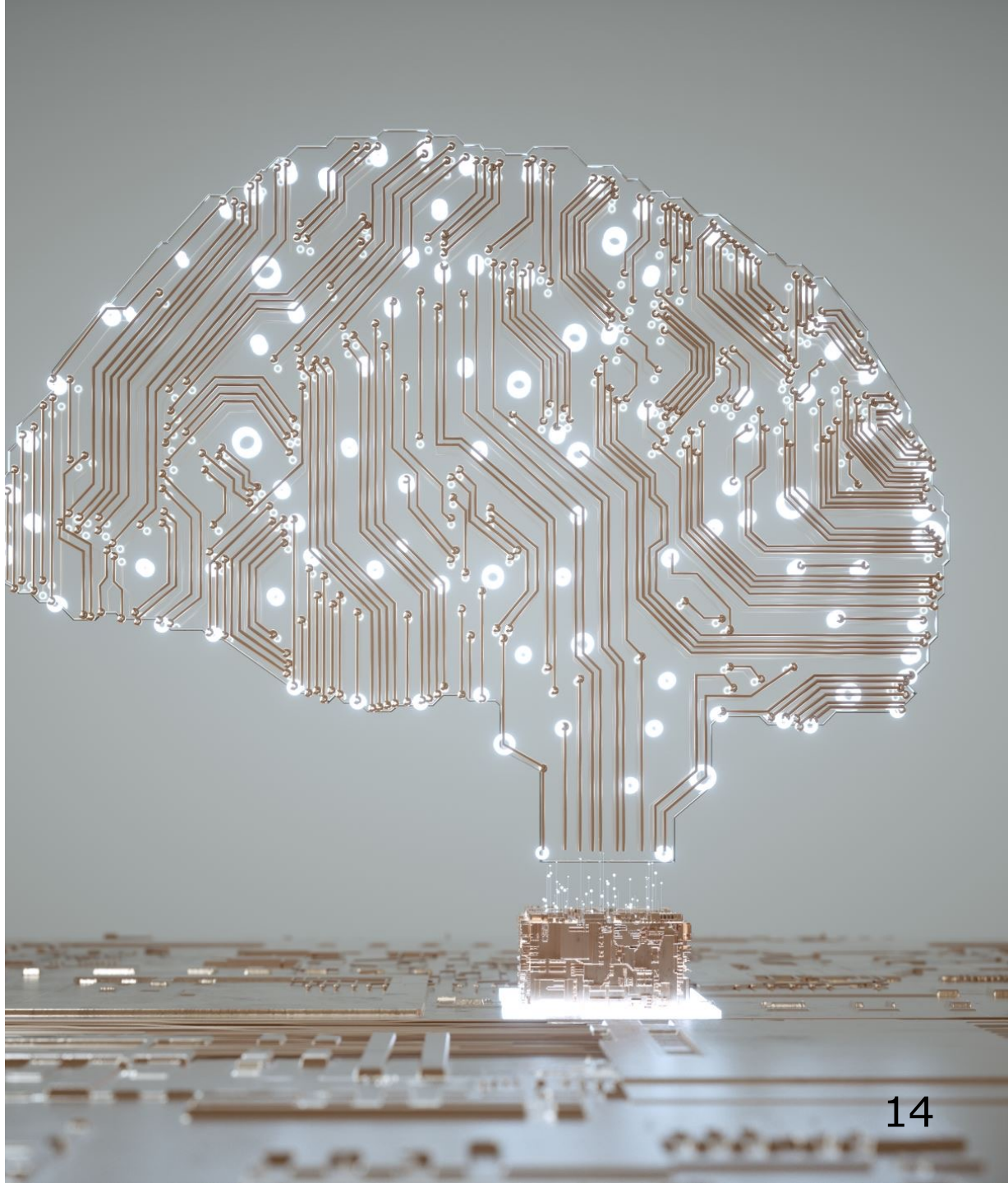
Advantages and Limitations of statistical models



AI-BASED MODELS

Why AI-based models?

Classical models struggle when systems become more complex, volatile, and non-stationary.

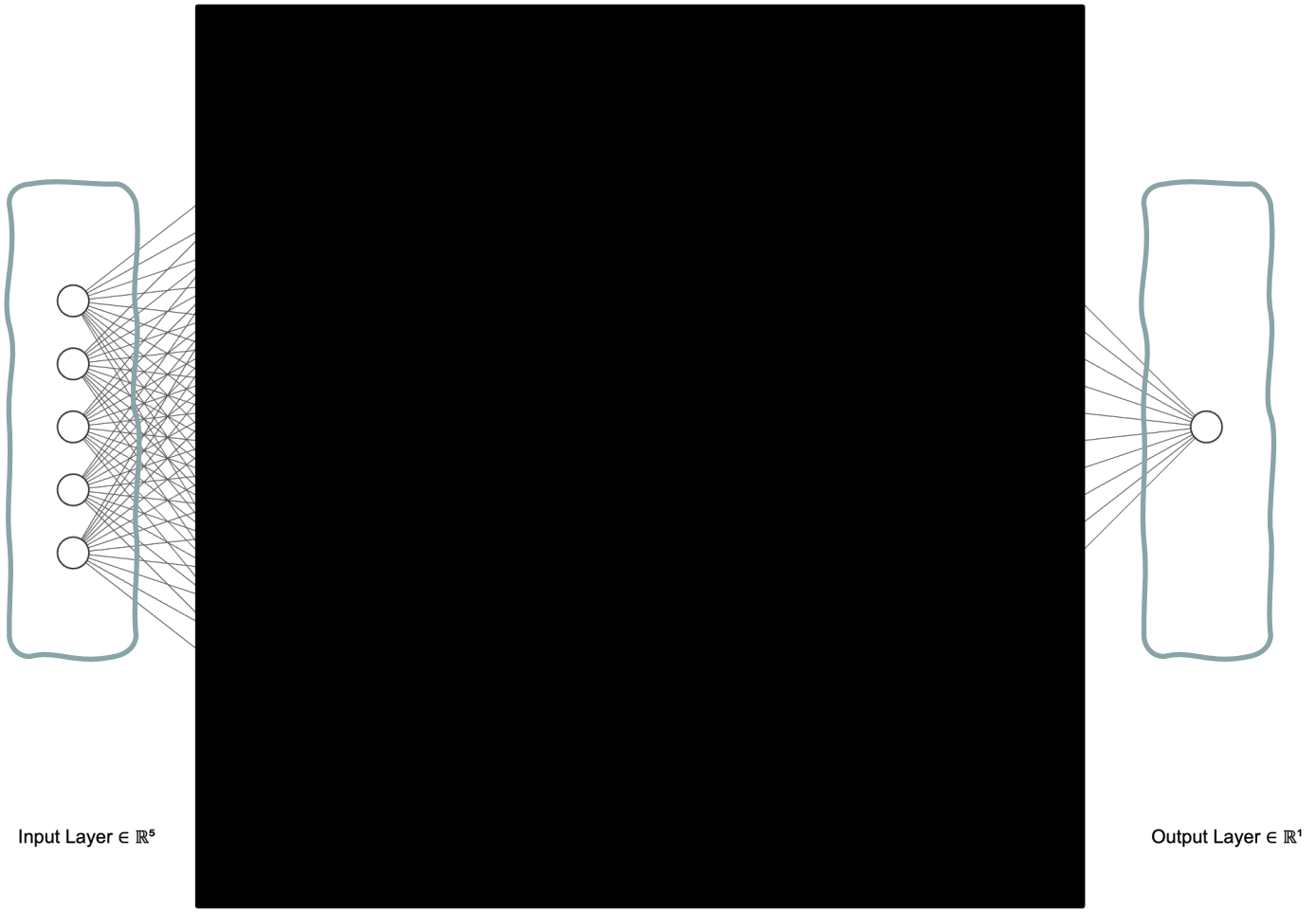


4TU.

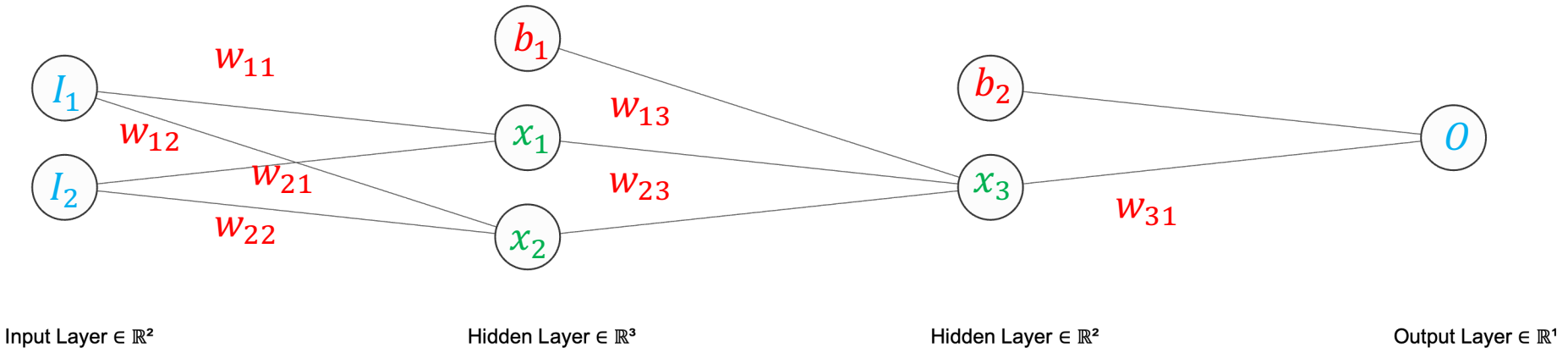
What is the first mathematical model that comes to your mind when thinking about AI models?

* No, I do not mean generative AI and therefore no ChatGPT, Claude, Gemini, etc.

Artificial Neural Network



A simple ANN



$$x_1 = f(I_1 \times w_{11} + I_2 \times w_{21})$$

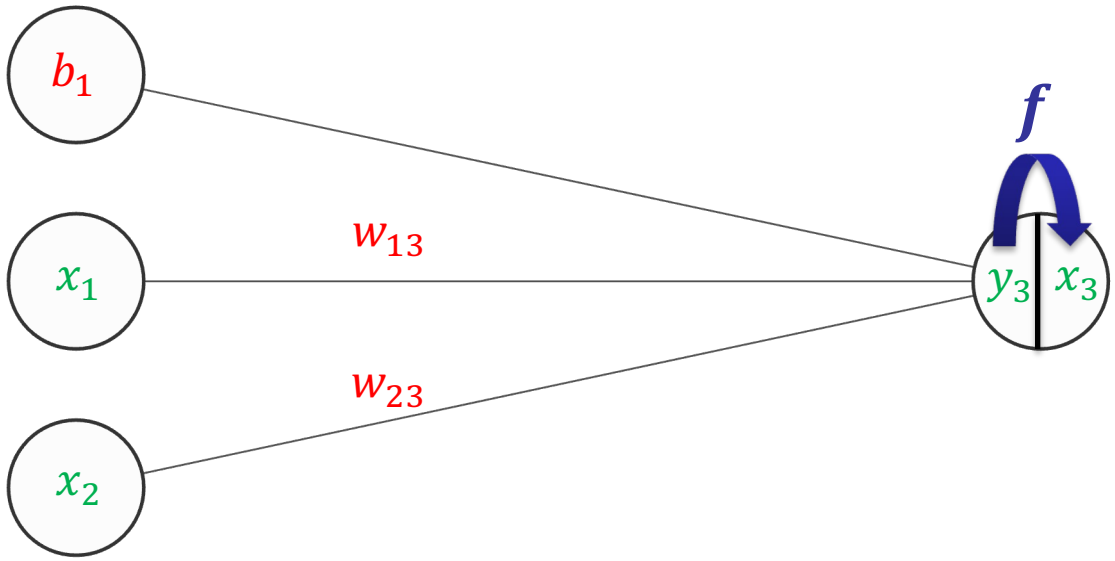
$$x_2 = f(I_1 \times w_{12} + I_2 \times w_{22})$$

$$x_3 = f(x_1 \times w_{13} + x_2 \times w_{23} + b_1)$$

$$O = f(x_3 \times w_{31} + b_2)$$

Training NN means learning weights (w) and biases (b)

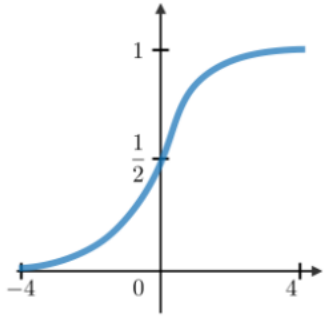
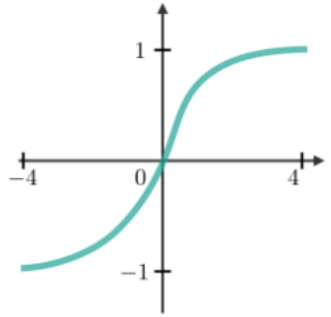
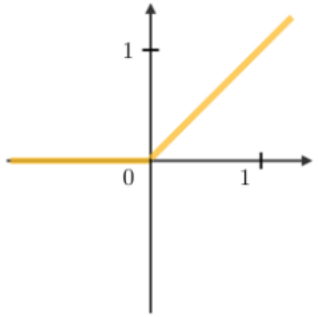
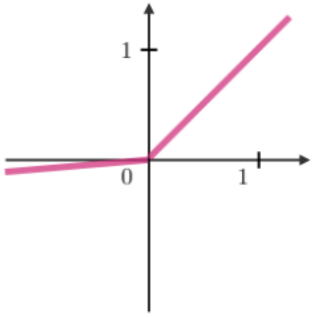
activation function



$$y_3 = x_1 \times w_{13} + x_2 \times w_{23} + b_1$$

$$x_3 = f(x_1 \times w_{13} + x_2 \times w_{23} + b_1)$$

Most common activation functions

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

credit: Medium

Activation function

$$x_3 = f(y_3)$$

$$y_3 = x_1 \times w_{13} + x_2 \times w_{23} + b_1$$

example: tanh function

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

how to calculate output using input?

- $O = f(x_3 \times w_{31} + b_2)$
- $O = \frac{e^{((\frac{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} - e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} + e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}) \times w_{31} + b_2)} - e^{-((\frac{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} - e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} + e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}) \times w_{31} + b_2)}}{e^{((\frac{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} - e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} + e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}) \times w_{31} + b_2)} + e^{-((\frac{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} - e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}{e^{(x_1 \times w_{13} + x_2 \times w_{23} + b_1)} + e^{-(x_1 \times w_{13} + x_2 \times w_{23} + b_1)}}) \times w_{31} + b_2)}}$

- $x_1 = \frac{e^{(I_1 \times w_{11} + I_2 \times w_{21})} - e^{-(I_1 \times w_{11} + I_2 \times w_{21})}}{e^{(I_1 \times w_{11} + I_2 \times w_{21})} + e^{-(I_1 \times w_{11} + I_2 \times w_{21})}}$
- $x_2 = \frac{e^{(I_1 \times w_{12} + I_2 \times w_{22})} - e^{-(I_1 \times w_{21} + I_2 \times w_{22})}}{e^{(I_1 \times w_{12} + I_2 \times w_{22})} + e^{-(I_1 \times w_{21} + I_2 \times w_{22})}}$
- $O = \ldots I_1 + \ldots I_2$
- we would like to learn Ws and bs to learn …

4TU.

Loss function

- A loss function maps model outputs and ground truth to a non-negative number that you minimize during training.
- It encodes what "good" means for the (AI) task.

- For data (x_i, y_i) and model $f_\theta(x_i)$

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(y_i, f_\theta(x_i)\right)$$

$$\theta \stackrel{\text{def}}{=} \{w, b\}$$

4TU.

Common loss functions

For deterministic tasks:

$$MSE = (y - \hat{y})^2$$

$$RMSE = \sqrt{(y - \hat{y})^2}$$

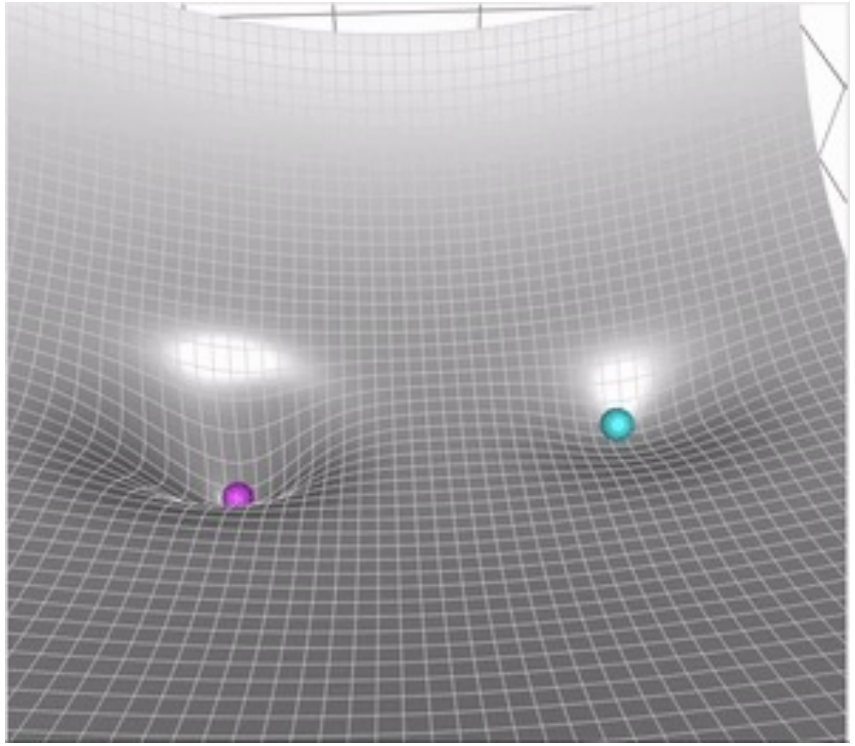
$$MAE = |y - \hat{y}|$$

For probabilistic tasks losses such as quantile or pinball loss can be used

$$L_{\tau(y,\hat{q})} = max\{\tau(y - \hat{q}), (\tau - 1)(y - \hat{q})\}$$

Optimizers

- An optimizer is the procedure that updates the model's parameters (weights and biases) to make the loss smaller. Gradient descent, momentum, etc.

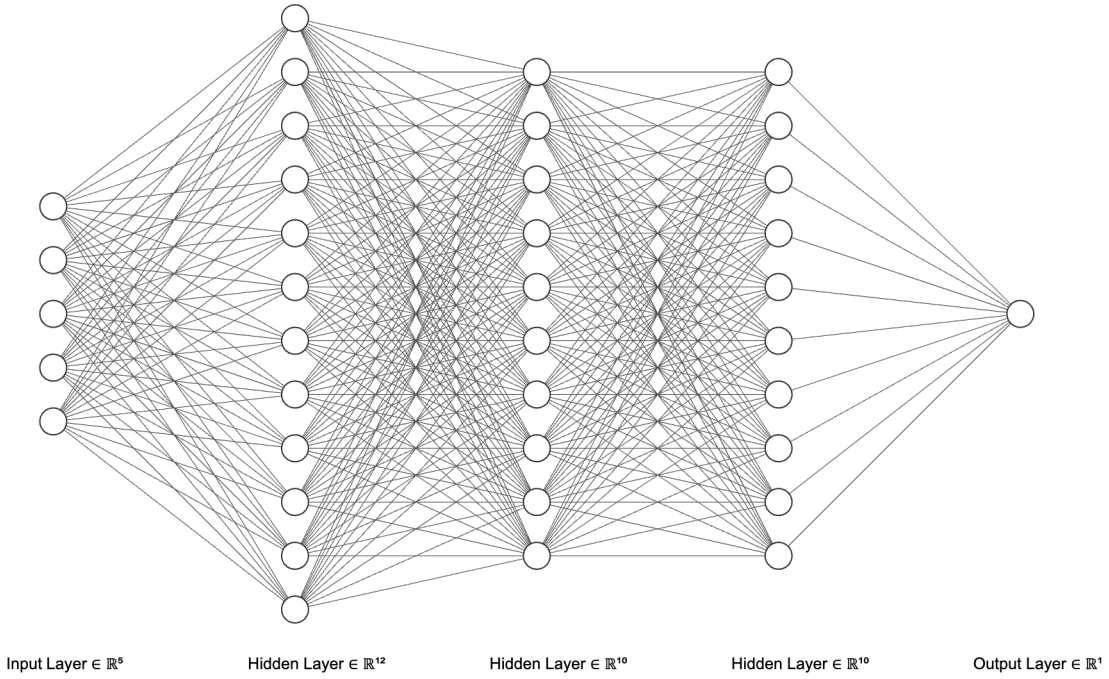


courtesy: https://medium.com/

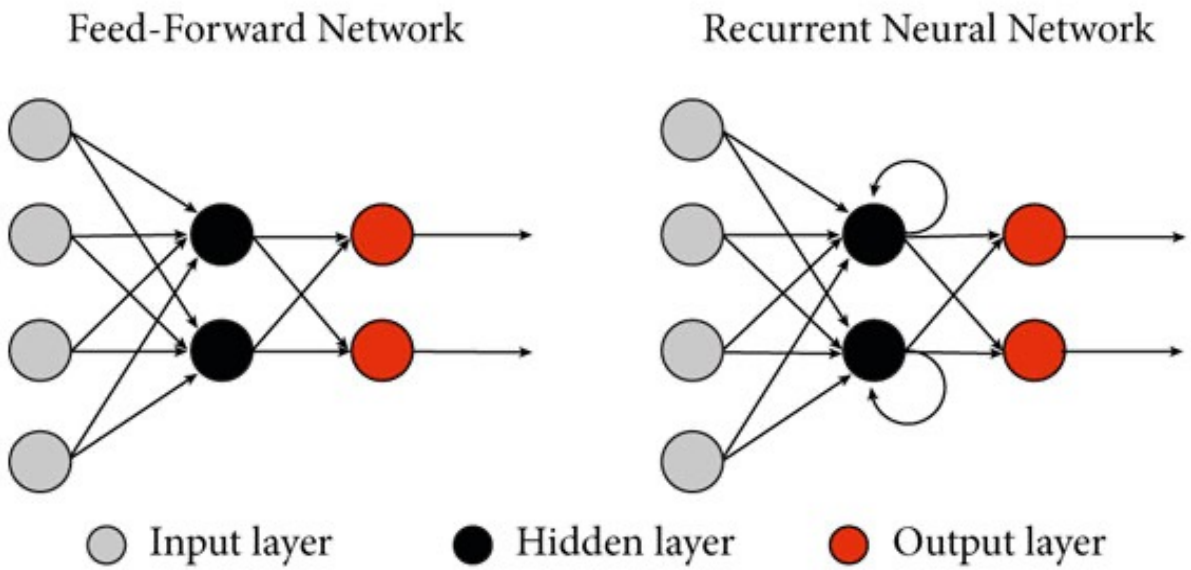
cyan: gradient descent
purple: momentum

NN architecture

- Feed-Forward Neural Network (FFNN)



Feed forward vs Recurrent



A standard FFNN processes each input independently and does not preserve temporal order or dependencies across time steps.

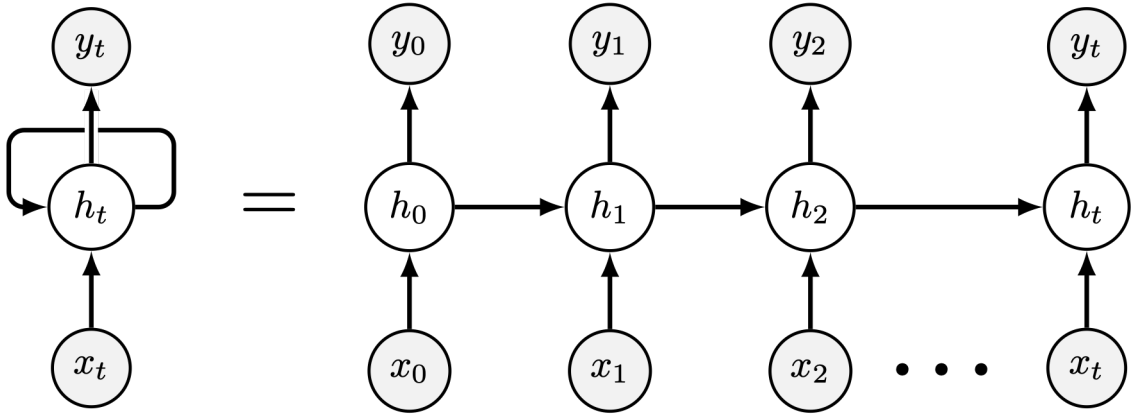
4TU.

Sequence matters!



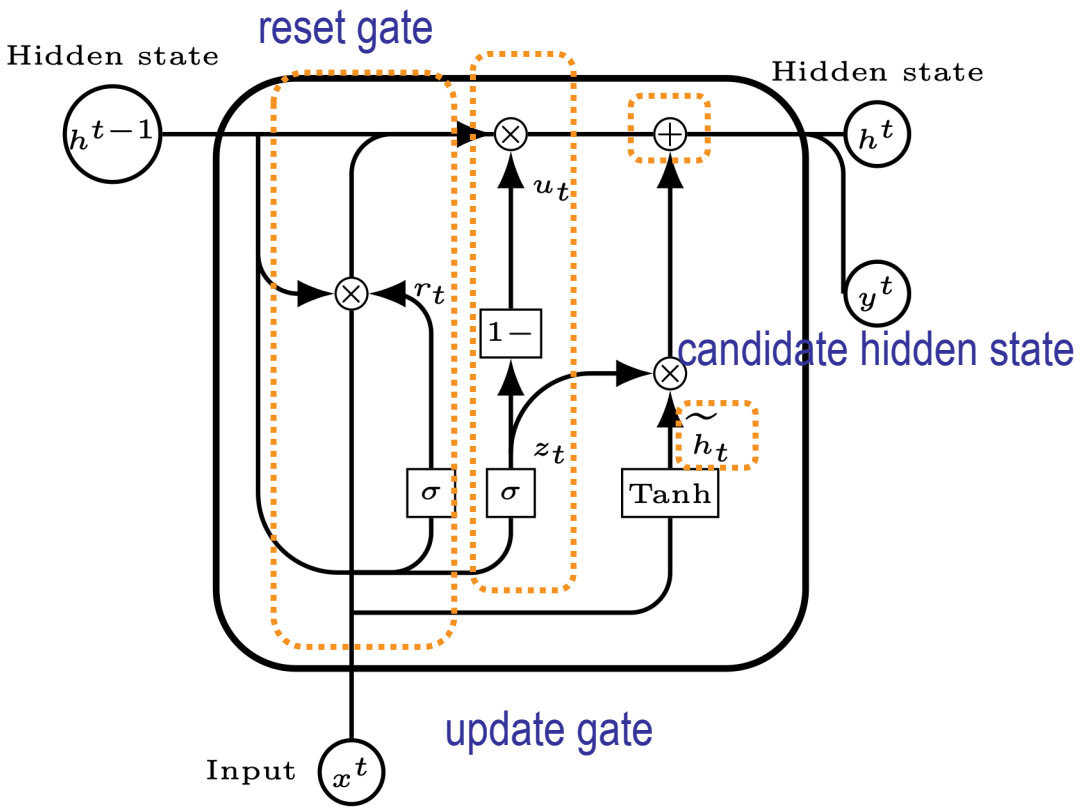
- **Sequence:** an ordered set of observations indexed over time.
- **Sequential learning:** learning temporal dependencies and patterns from sequence
- **Mechanisms**: 1) Memory, 2) convolution, 3) attention

Recurrent Neural Network (RNN)



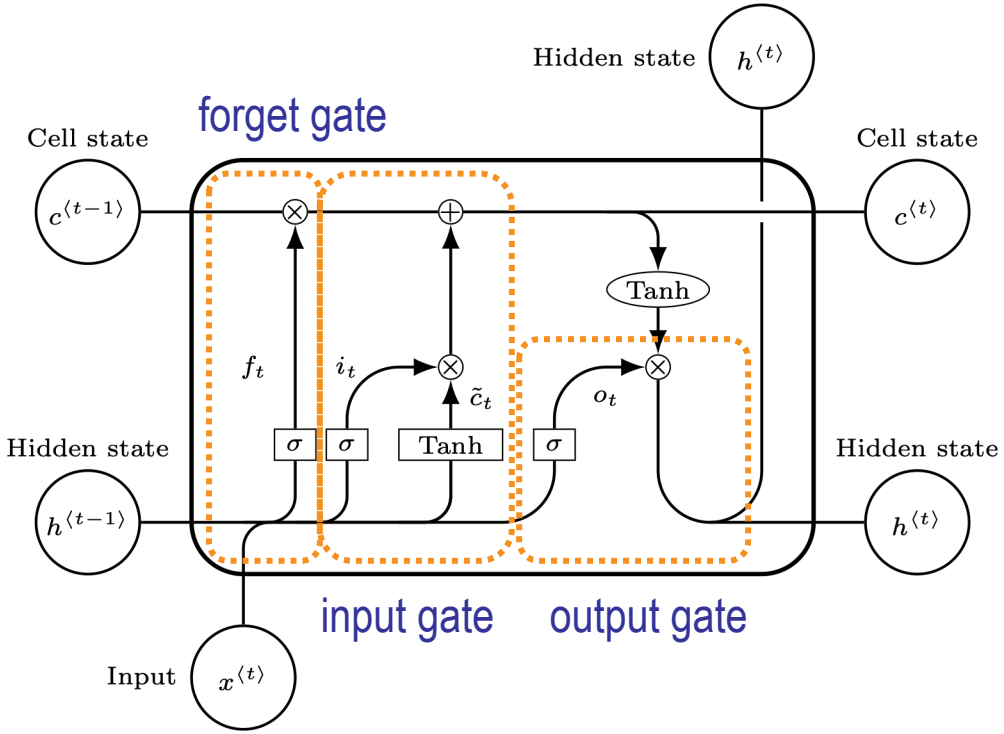
- has a memory
- but memorizes it all!

Gated Recurrent Unit (GRU)



- has a memory
- doesn't memorize it all!
- but only has a short-term memory (h)

Long-Short Term Memory (LSTM)



- has a memory
- doesn't memorize it all!
- has both long-term (c) and short-term (h) memory

Convolution Mechanism

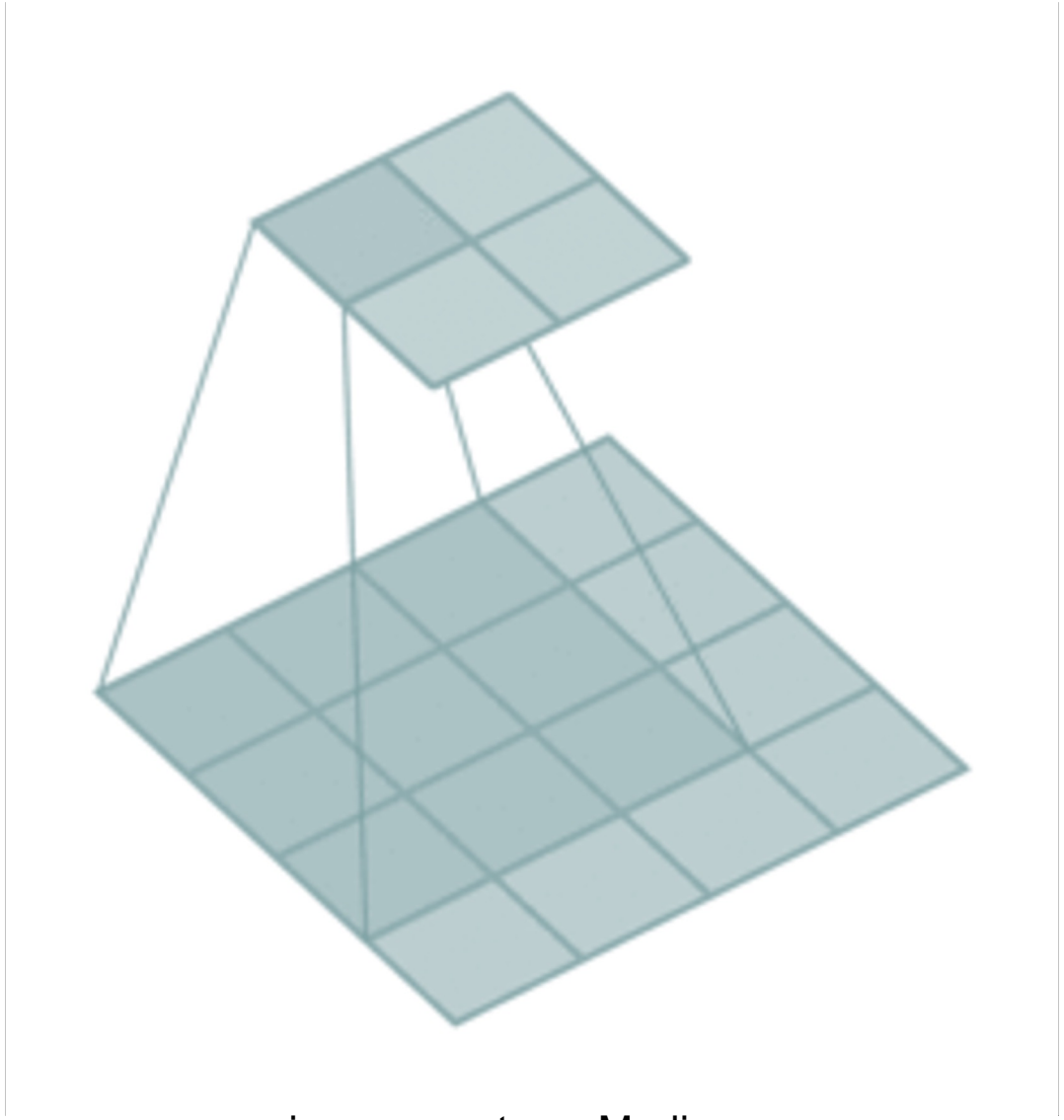
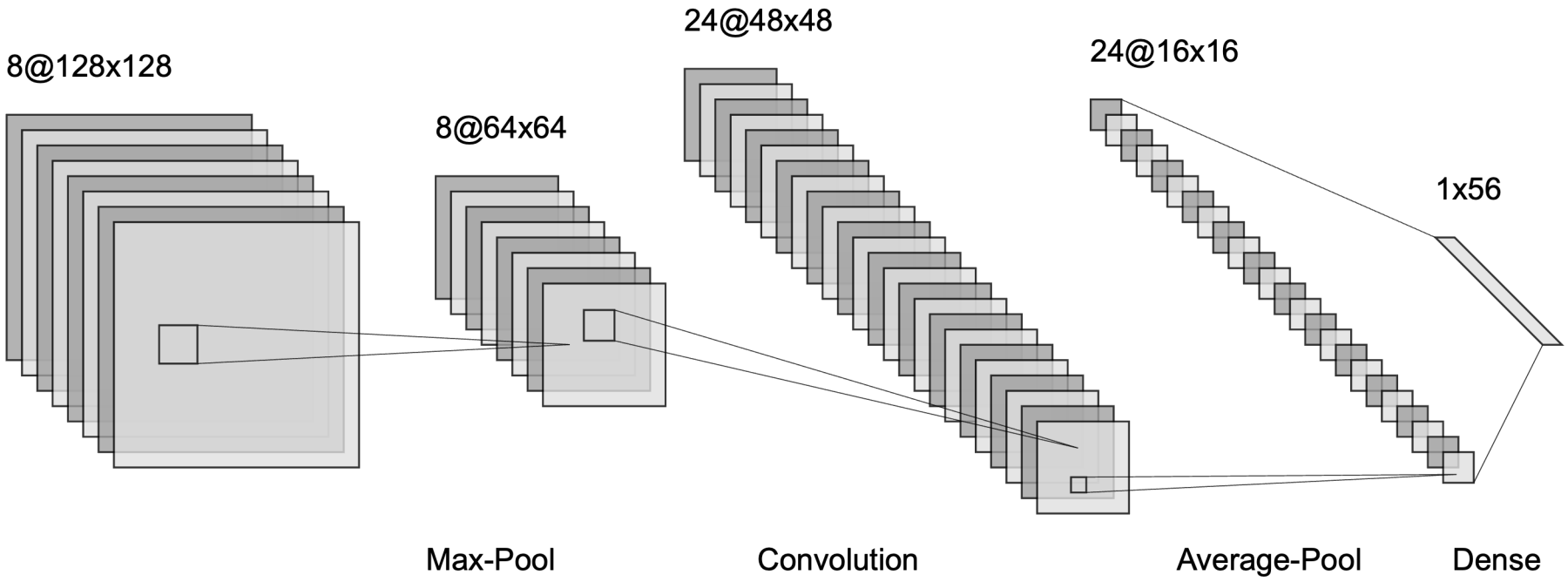


image courtesy: Medium

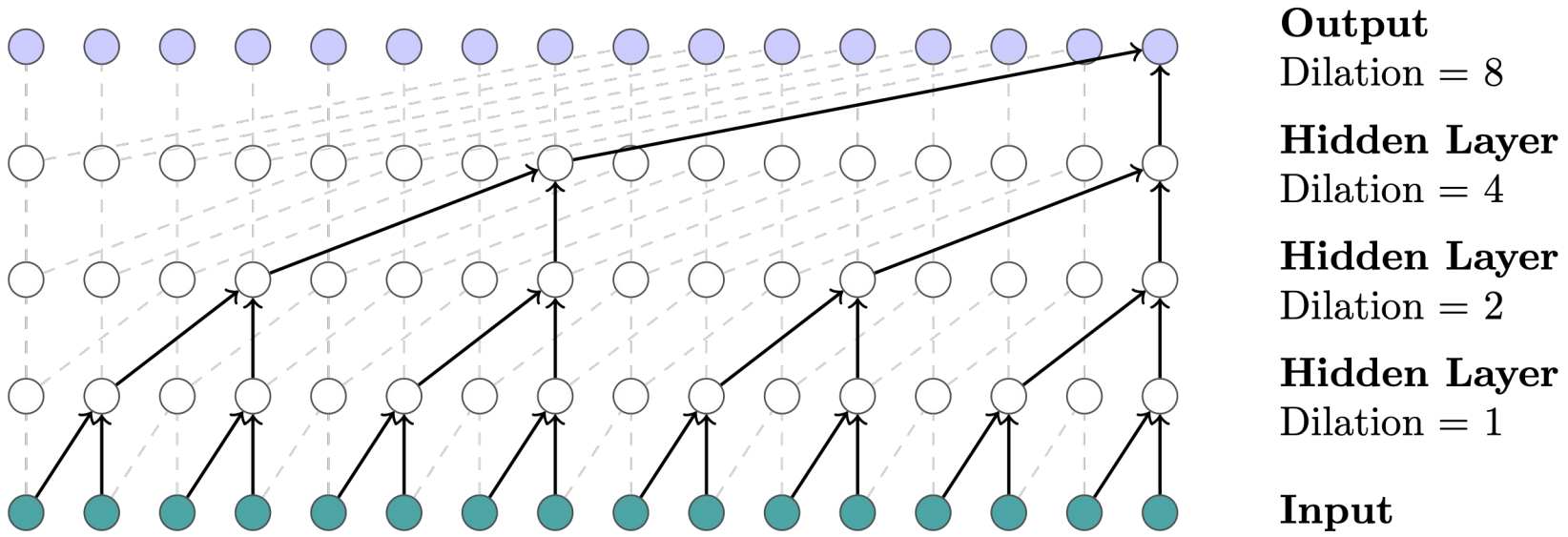
Convolutional Neural Network (CNN)



Temporal Convolutional Network (TCN)

TCNs are CNNs redesigned for sequential time-series forecasting. TCN captures longer temporal dependencies compared to CNN:

1. Causal convolution
2. Dilated convolution

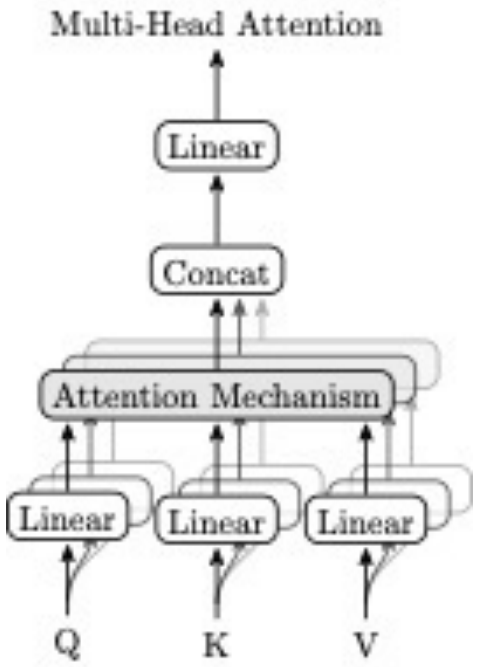


Attention mechanism

Attention mechanism: The model looks at the sequences and decides which parts are important.

Q: “What information do I need?”
K: “What information does this time step contain?”
V: “The useful information itself.”

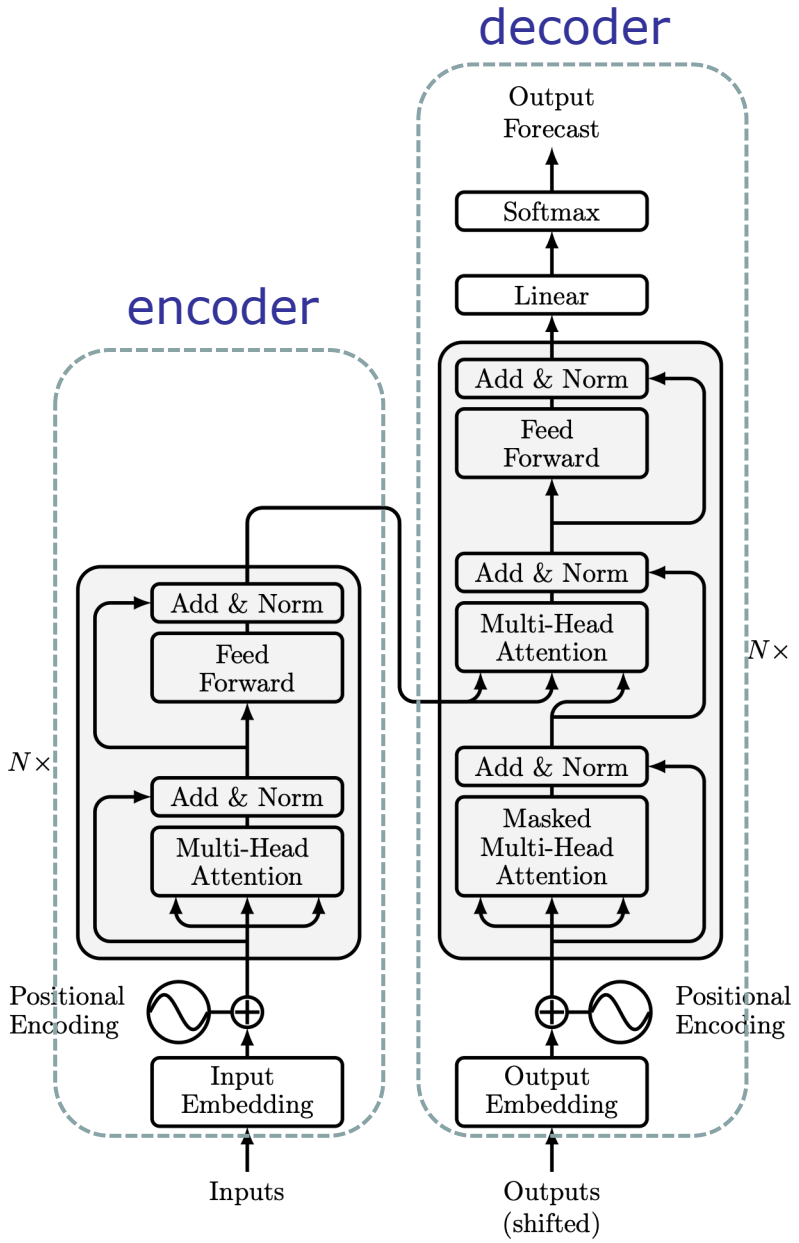
$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Transformer

Encoder-decoder architecture:
the encoder learns sequence representations, and
the decoder maps them to the target output sequence.

Attention helps the model focus on the most relevant time steps



Case study – SlimPark

#	Hardware	Model	Capacity
69	PV panels	Perlight PLM-360-72DG	360 Wp
1	PV inverter	SolarEdge SE25K	25 kW
1	Energy storage	Kiwatt Power Pack 15 kW inverter	30 kWh
9	EV chargers AC	Mennekes Amtron Professional+	22 kW



Training dataset: 1-9-2021 until 26-05-2023
Test dataset: 27-05-2023 until 31-10-2023

$$\text{nRMSE} = \frac{\sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_i - \hat{y}_i)^2}}{\max(y) - \min(y)} \times 100\%$$

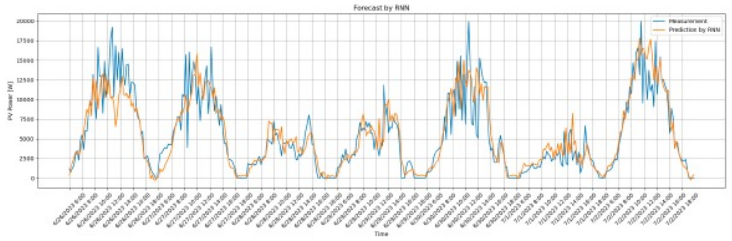
$$\text{nMAE} = \frac{\frac{1}{N}\sum_{t=1}^{N}|y_i - \hat{y}_i|}{\max(y) - \min(y)} \times 100\%$$

4TU.

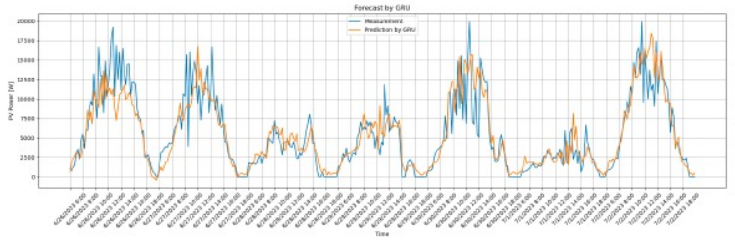
Hyperparameter and configuration of the models

Model	Architecture	Unit	Kernel size	Dropout	Optimizer	Activation	Learning rate
RNN	Vanilla RNN + D	64,64		0.3	ADAM	ReLU	0.001
GRU	GRU+D	64,64		0.3	ADAM	ReLU	0.001
LSTM	2LSTM+D	64,64		0.3	ADAM	ReLU	0.001
CNN	2Conv 1D+Max Pool + D	64,128,64	3	0.3	ADAM	ReLU	0.001
TCN	Causual Conv 1D + Dilated Causal Conv 1D + D	64,64,64	3	0.3	ADAM	ReLU	0.001
Transformer	MHA + FFN + D	2,56,64		0.3	ADAM	ReLU	0.001

Day-ahead PV forecasting results



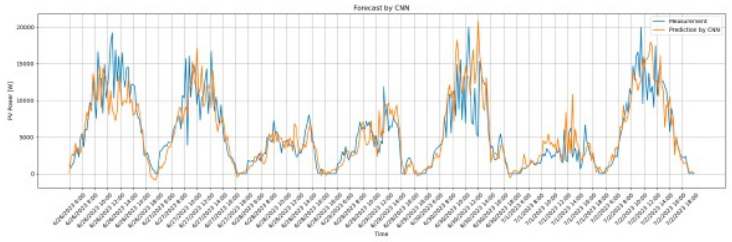
(a) RNN



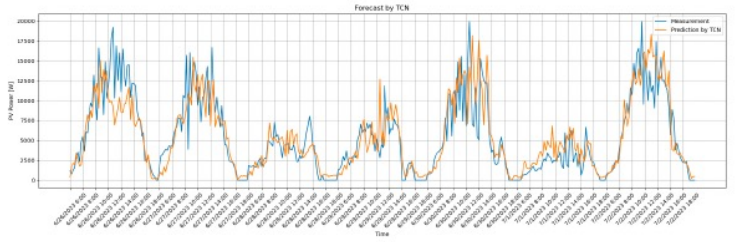
(b) GRU



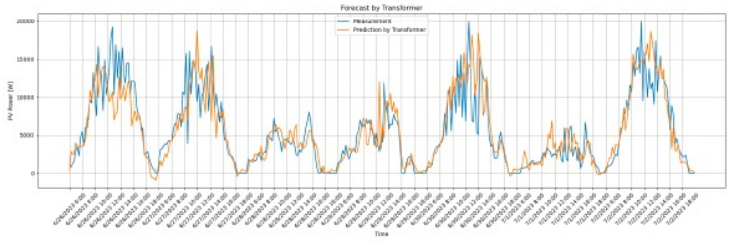
(c) LSTM



(d) CNN



(e) TCN



(f) Transformer

Day-ahead PV power forecasting errors

Method	nRMSE	nMAE
RNN	10.38%	6.54%
GRU	10.45%	6.47%
LSTM	10.60%	6.50%
CNN	11.25%	6.73%
TCN	11.92%	7.7%
Transformer	10.62%	6.89%
Persistence	20.01%	11.97%

4TU.

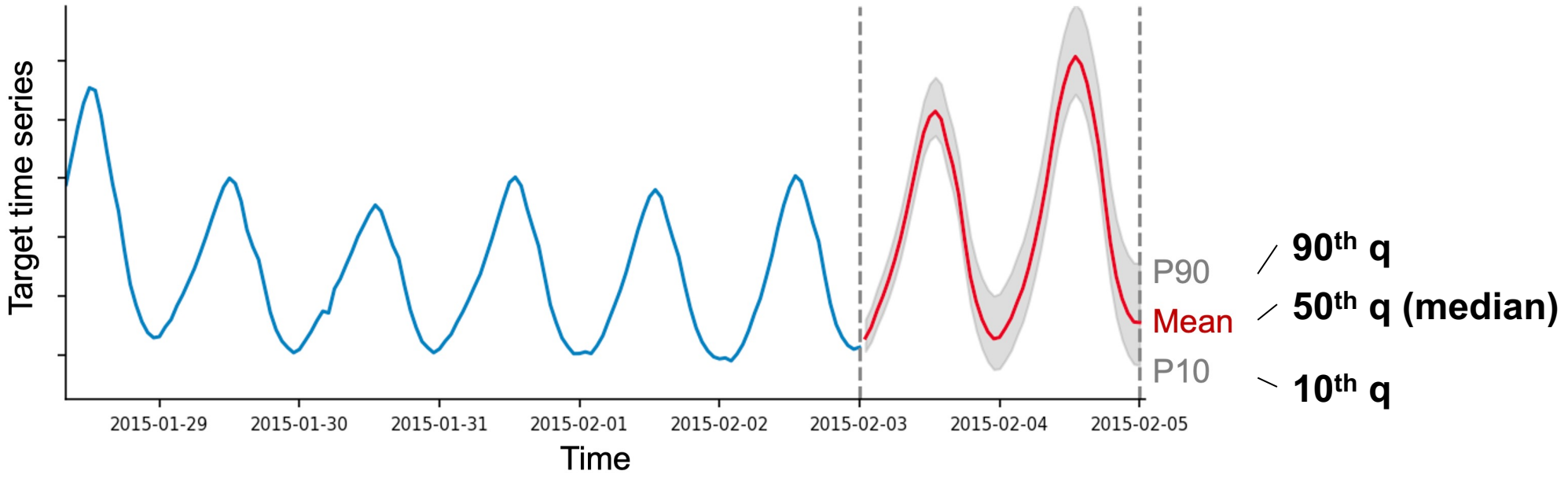
Every flower has a smell

Method	Best for	Limitations
RNN	Simple sequential patterns with short-term dependencies	Struggles with long-term dependencies; vanishing gradient problem
GRU	Efficient training; moderate sequence lengths	underperforms LSTM on tasks requiring complex memory management
LSTM	Long-term dependencies; robust sequence learning	Computationally intensive; slower training
CNN	Local temporal patterns; fast training	Limited in capturing long-range dependencies
TCN	Long-range temporal dependencies, efficient parallel training, stable sequence modeling	Higher computational cost, sensitive to dilation/kernel configuration
Transformer	Very long sequences; learning complex dependencies, parallel training	Requires large datasets; high computational cost
Persistence	Very short term forecasts, benchmark	Naïve!

PROBABILISTIC FORECAST

Quantile Regression

Quantile regression is an extension of linear regression used to estimate the conditional median or any other specific quantiles (such as the 10th or 90th percentiles) of a variable.



a.s.r. Smart Car Park

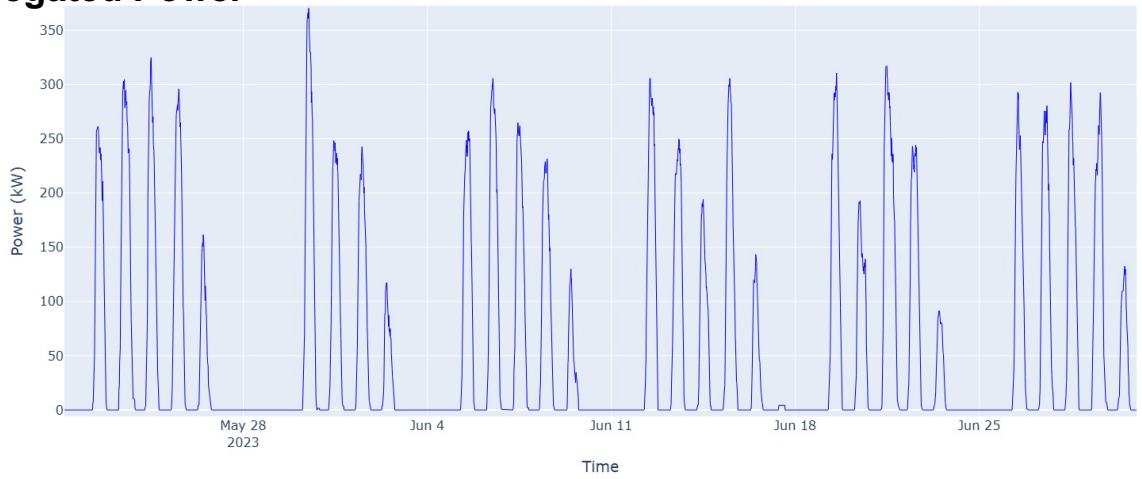




- EU's largest EV charging station (2021) with 250 smart bi-directional charging stations, Utrecht, The Netherlands
- A roof with more than 2,000 solar panels which can provide for 4 million electric kilometres a year.

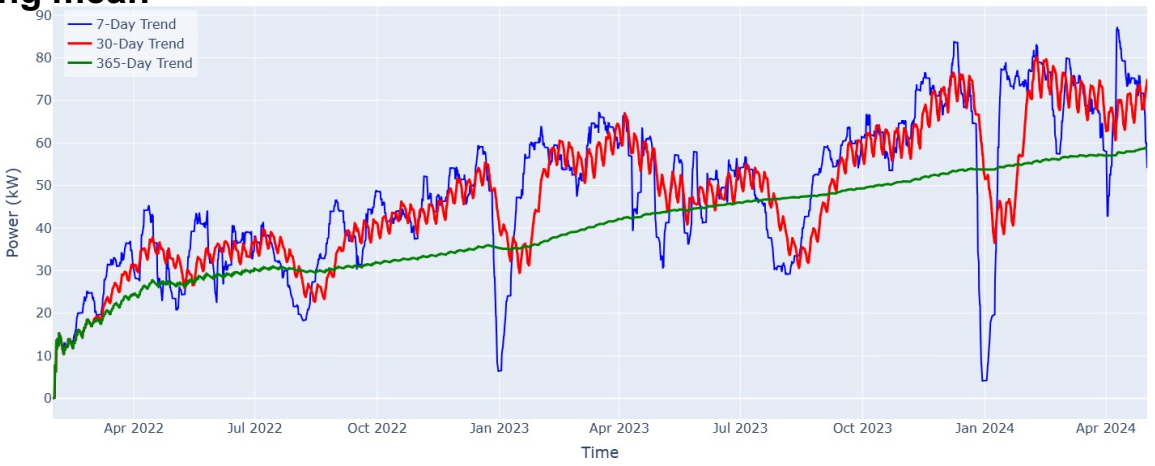
EV demand analysis

Aggregated Power



- The data spans from August 25, 2020, to October 2, 2024.
- We decided to keep data from January 2023 to September 2024

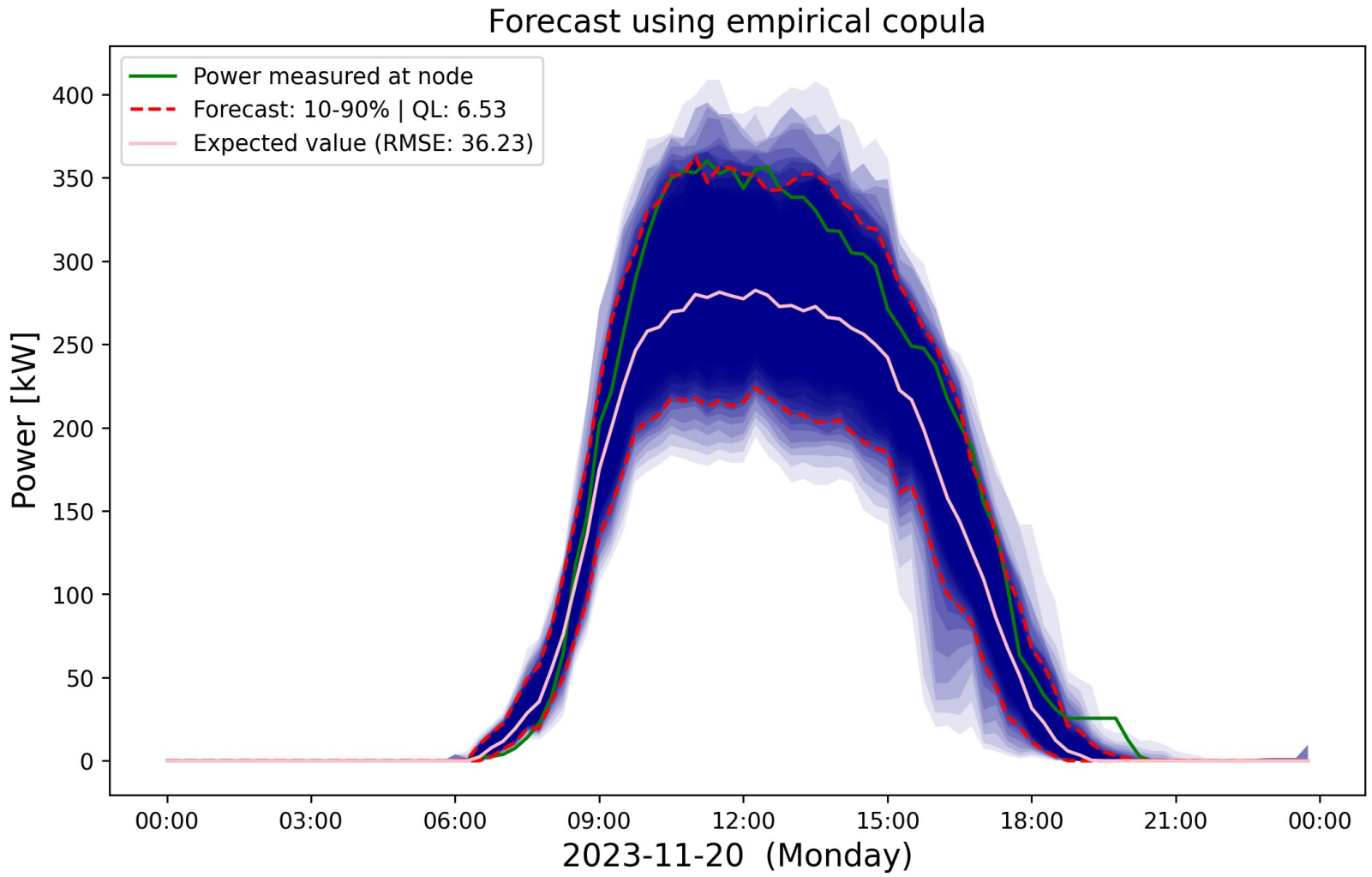
Rolling mean



Increasing trend in the dataset due to:

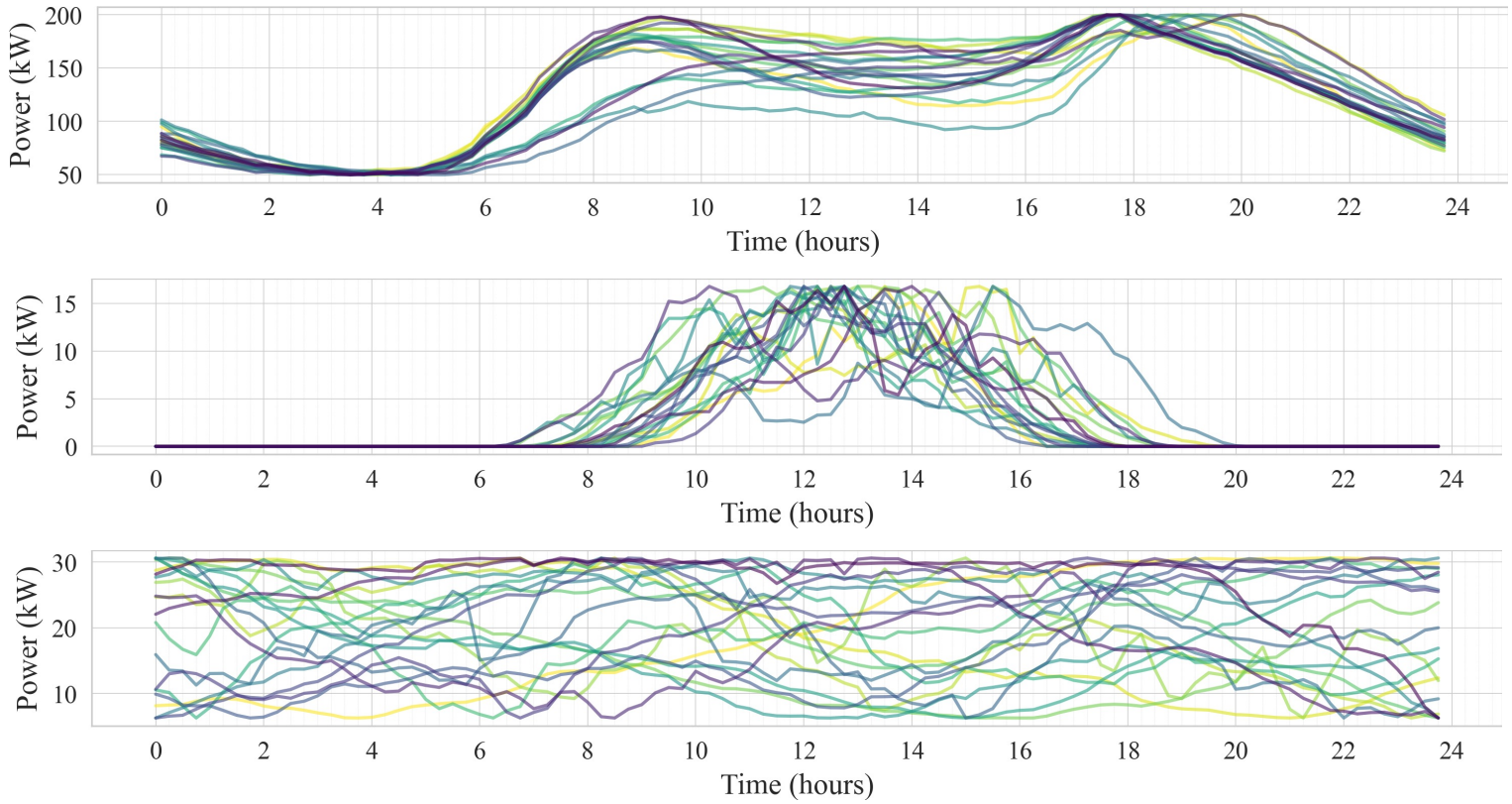
- New charging points
- Larger EV fleet
- COVID restrictions

EV demand forecast



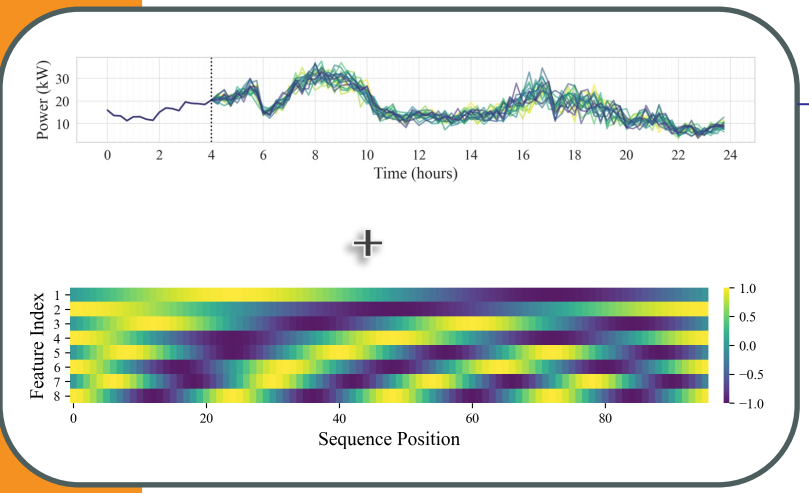
DO WE NEED A PERFECT FORECAST?

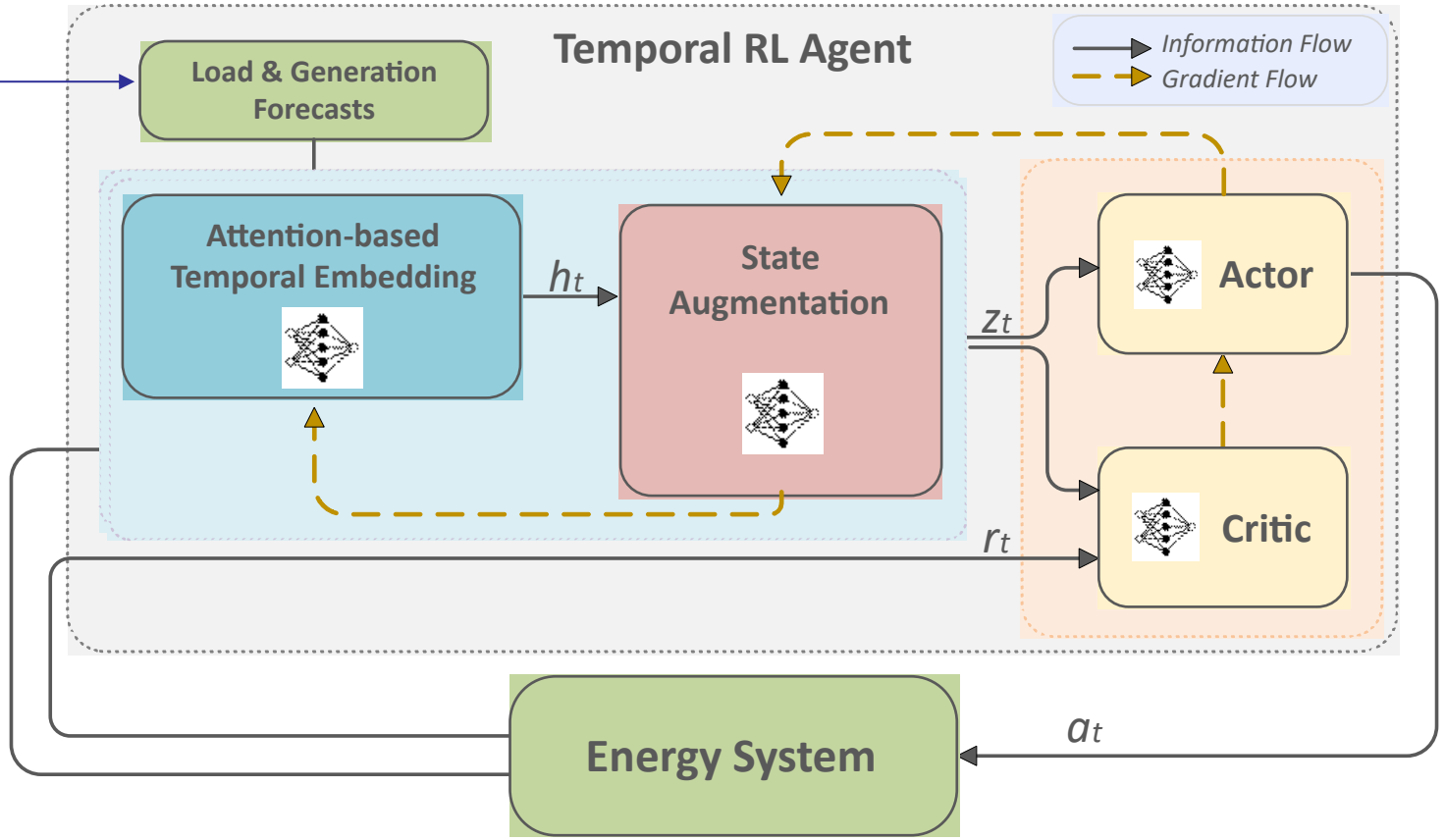
Variable inputs



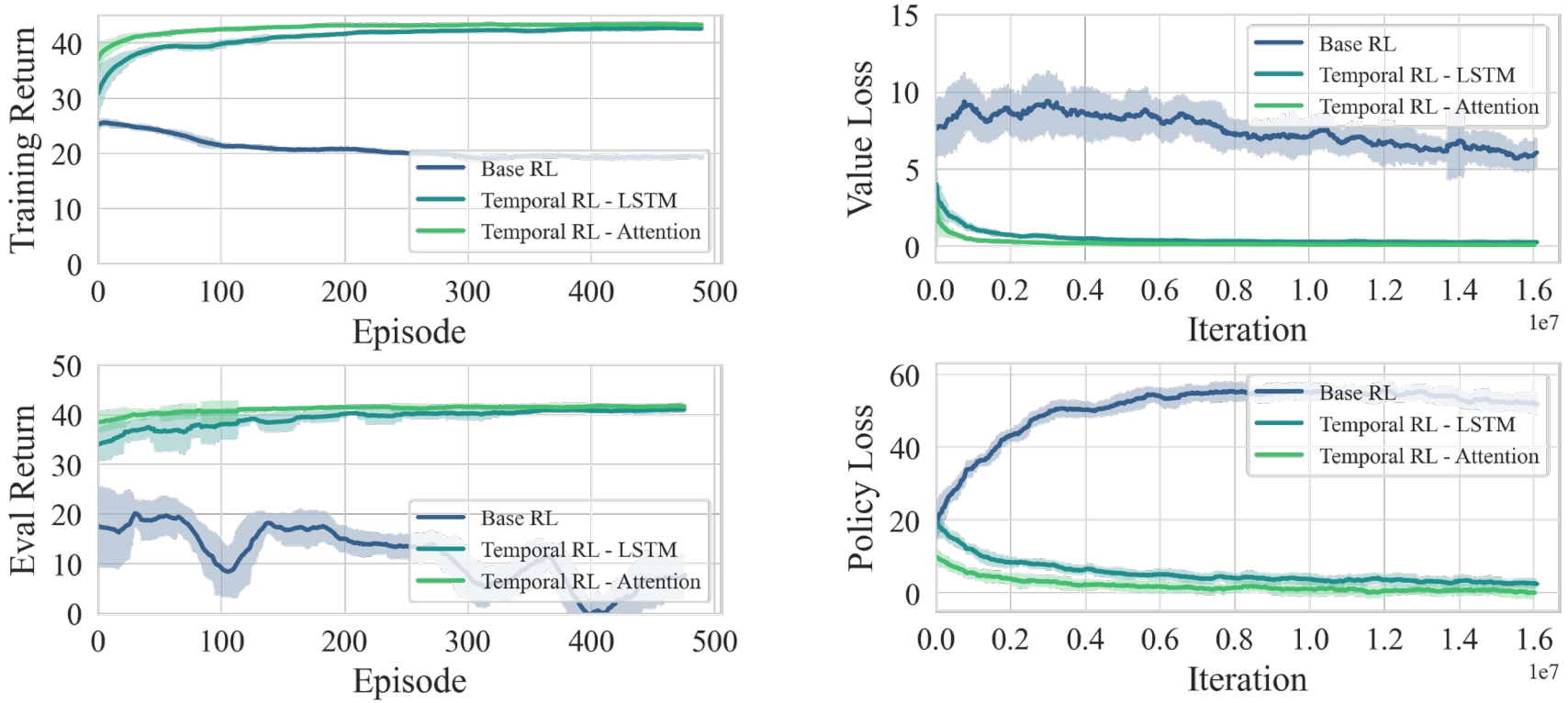
Temporal Awareness

Temporal RL - Agent





Results – Agent Training



Training results of different RL agents.

Code

In case you would like to check the code:

1. https://github.com/AIER-Lab/Solar-Fork
2. https://github.com/AIER-Lab/ES-Temporal-RL

Solar Forecast

This is a work in progress ... The repo includes sequential learning models for solar PV forecast; model tuning,... we keep updating the repo

Repo Structure

```
solar_fork/
├── models/
│   ├── forecast.py
│   │   ├── Class: ForecastModel
│   │   │   ├── Function: __init__()
│   │   │   ├── Function: build_model()
│   │   │   ├── Function: train()
│   │   │   ├── Function: predict()
│   │   │   ├── Function: save_model()
│   │   │   ├── Function: load_model()
│   ├── nn.py
│   │   ├── Function: build_rnn_model()
│   │   ├── Function: build_gru_model()
│   │   ├── Function: build_lstm_model()
│   │   ├── Function: build_cnn_model()
│   │   ├── Function: build_tcn_model()
│   │   ├── Function: build_transformer_model()
│   │   ├── Function: build_ann_model()
├── utils/
│   ├── assets.py
│   │   ├── Function: evaluate_and_plot()
└── data/
    ├── dataset.py
    │   ├── Class: SeqDataset
    │   │   ├── Function: __init__()
    │   │   ├── Function: normalize_df()
    │   │   ├── Function: create_sequences()
    │   │   ├── Function: split_data()
    ├── data_UT_clean.csv
```

It's not magic …

Next time someone asks you about predicting future, you'll know:

It's math + models !



4TU.

Thank you for your “QKV”!

QUESTIONS?

Send me an email: e.shirazi@utwente.nl

TU Delft Delft University of Technology

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF TWENTE.

WAGENINGEN UNIVERSITY & RESEARCH

