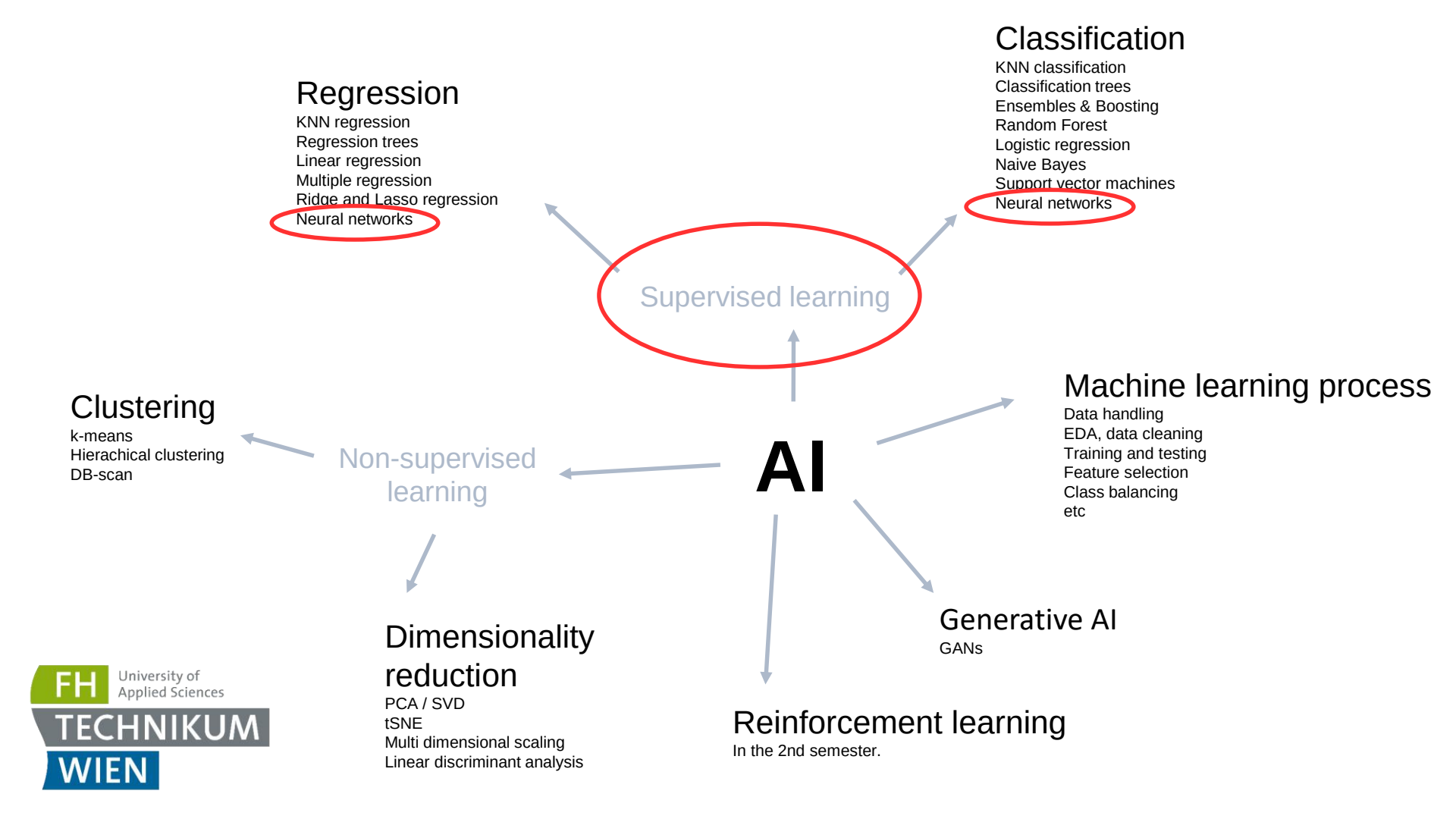
# Recurrent Neural Networks (RNNs)

ML2: AI Concepts and Algorithms (SS2025)
*Faculty of Computer Science and Applied Mathematics*
*University of Applied Sciences Technikum Wien*

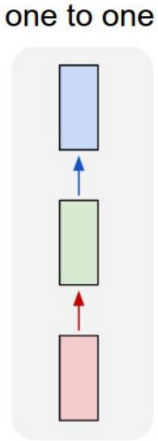
FH University of Applied Sciences TECHNIKUM WIEN

**Lecturer:** Rosana de Oliveira Gomes
**Author:** B. Knapp, S. Rezagholi, R.O. Gomes




FH University of Applied Sciences TECHNIKUM WIEN

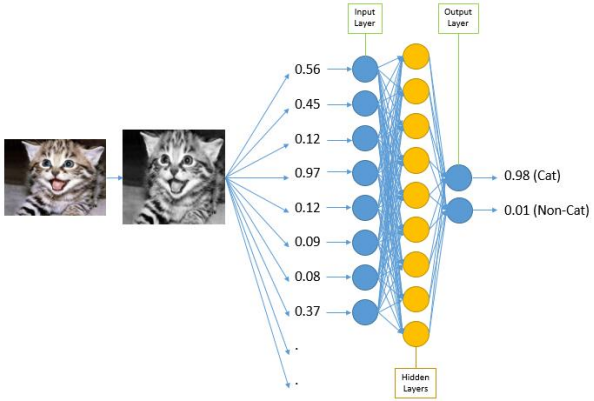
# Sequential Problems




Typical ANN: input layer → hidden layer → output layer

ML Problems: Regression, Classification

**One to one relationship between input and output**




FH University of Applied Sciences TECHNIKUM WIEN

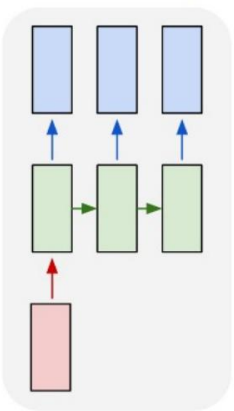
# Sequential Problems

**What if the input-output relationship is different?**

Sequential data: relationship in the sequential order of data points (e.g: tabular, text, audio, video)

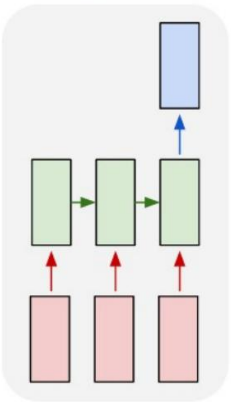
Image captioning

one to many




Review Ratings

many to one

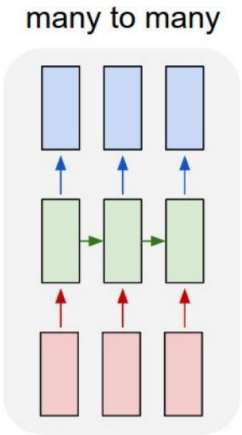




FH University of Applied Sciences TECHNIKUM WIEN

# Sequential Problems

**Sequence to Sequence (seq2seq):**
takes multiple input and gives multiple outputs

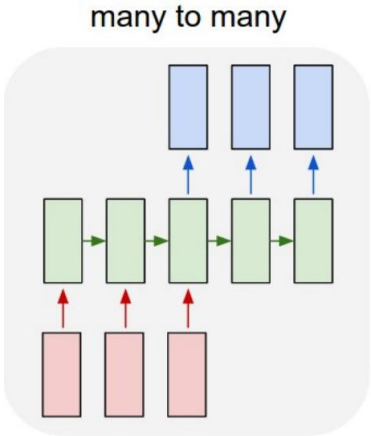



## Named Entity Recognition (NER)

Input and output are sequences of words with **same lengths**.

Input: `["Einstein", "was", "born", "in", "Germany"]`

Output: `["PERSON", "O", "O", "O", "LOCATION"]`




## Machine Translation

Input and output are sequences of words with **different lengths**.

**Input (German, 9 words):**

`"Obwohl es schon spät war, ging er nach Hause."`

**Output (English, 11 words):**

`"Although it was already late, he went home."`

FH University of Applied Sciences TECHNIKUM WIEN

# Recurrent Neural Networks (RNNs)

- Multilayer perceptons have poor performance for sequential data (and time series).
- In sequential data 1-step predictions are often not sufficient. A larger segment of the past is necessary for a robust forecasting.
- Nontrivial time series exhibit high degrees of dependence between the values of variables at different time points (at least for moderate time differences).
- **RNNs are neural networks optimized for sequential data, able to store information about past steps.**
- Example applications:

  Stock price data (time series),
  Natural language (sequential data),
  Audio/Image/Video Captioning (Computer Vision)

# Quiz Time

A **medical AI system** is designed to analyze a sequence of **heartbeat signals** from an electrocardiogram (ECG) and:

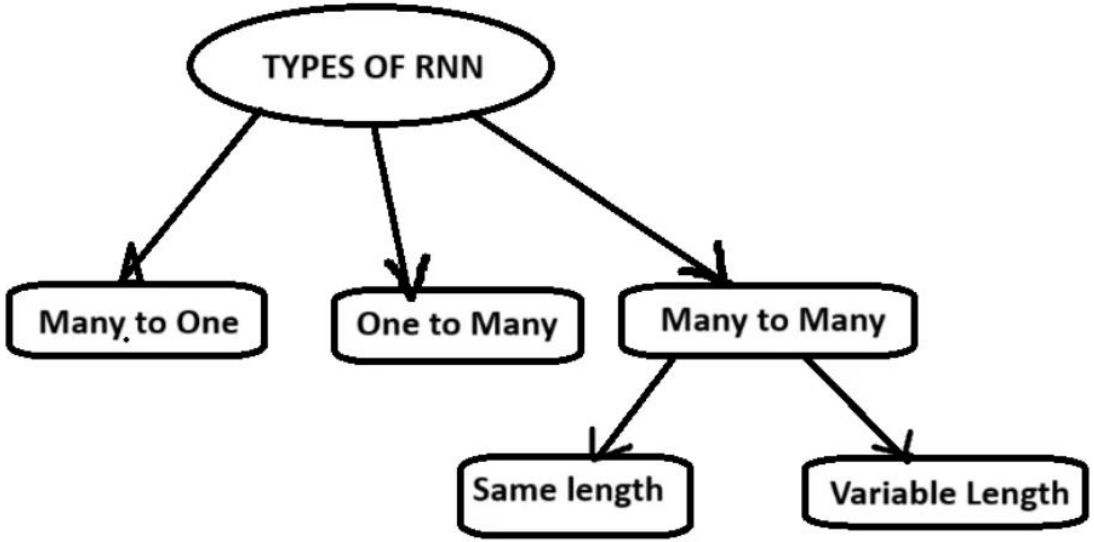
**Detect whether the patient has an abnormal heart condition (binary classification). If an abnormality is detected, generate a sequence of possible risk factors based on past medical history.**

Which type of sequential processing best describes this problem?

**A)** Many-to-One followed by One-to-Many
**B)** One-to-Many followed by Many-to-One
**C)** Sequence-to-Sequence (varying input/output length)
**D)** Many-to-Many (fixed input/output length)

FH University of Applied Sciences TECHNIKUM WIEN

# RNNs: Sequential Processes

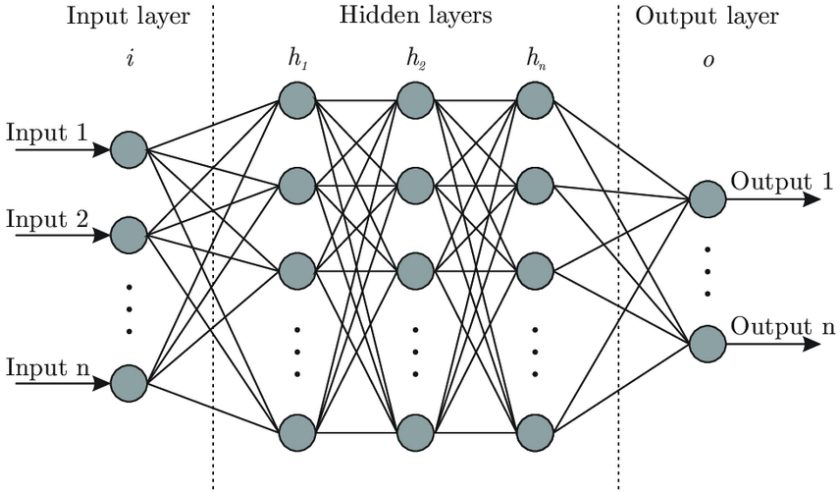



**How to store information of previous steps into a neural network?**

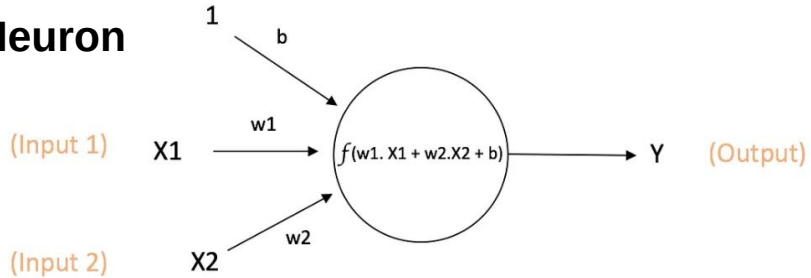
FH University of Applied Sciences TECHNIKUM WIEN

# Recap: ANNs

**Architecture:** input, hidden, output layers




**Neuron**




**Forward propagation:** prediction, Loss

**Back Propagation:**
update weights to minimize loss

**Iterative gradient descent training algorithm.**

Initialize $w, \eta$

Compute $out, E_{tot}$

$\forall w$ :

$$\Delta w = -\frac{\delta E_{tot}}{\delta w}(1)$$

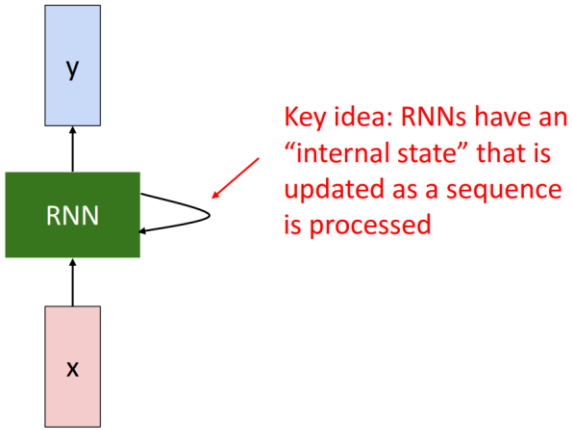
$$w_{new} \leftarrow w_{old} + \eta \Delta w + ...(2)$$

Compute $out, E_{tot}$

Repeat until $E_{tot} < \epsilon$

FH University of Applied Sciences TECHNIKUM WIEN

# RNNs: Formulation




We can process a sequence of vectors **x** by applying a **recurrence** formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state

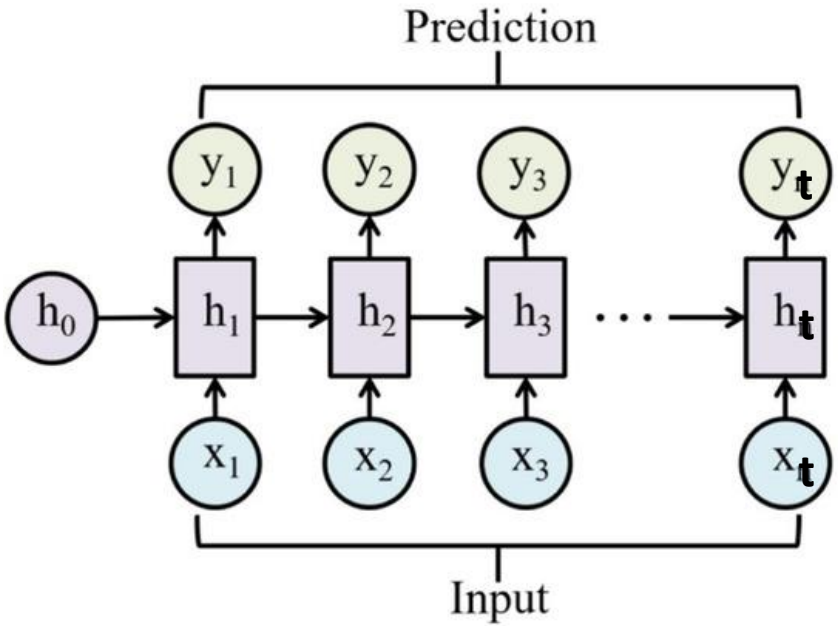
some function with parameters W

old state

input vector at some time step

The W matrix has always the same parameters in every time step. Same weights to process every point in time.

# RNNs: Architecture




**„Vanilla Recurrent Neural Networks**

The state consists of a single *"hidden"* vector **h**:

$$h_t = f_W(h_{t-1}, x_t)$$

(also bias term)

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

$$y_t = W_{hy}h_t$$

**Weight matrices**

**(h0, x1) → (h1, y1)**

**…**

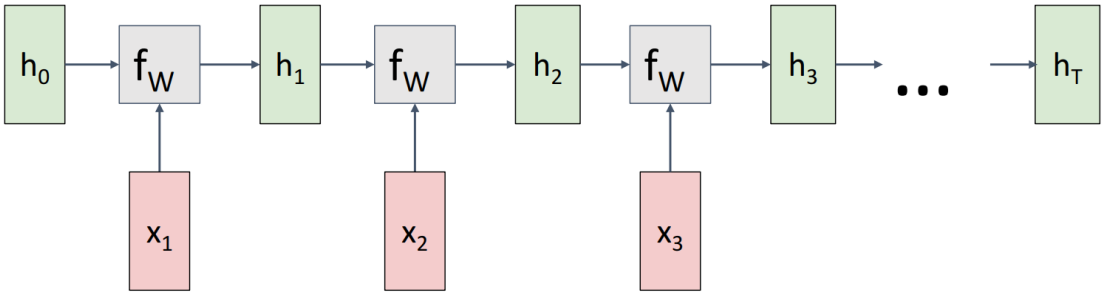
**(ht-1, xt) → (ht, yt)**

[Finding Structure in Time. Elman (1990)]

# RNNs: Unrolling

Training an RNN requires unrolling it for „backpropagation in time”.

- Initial hidden state **h0** (often set to zeros)
- **Same weight matrix in all steps**



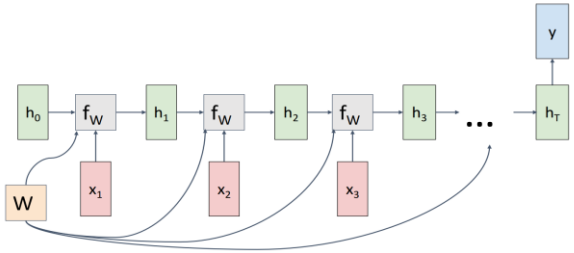

**The unrolled network is not a multilayer perceptron:** The recurrent structure shows up via the multiple appearance of certain weights and biases (W matrices).

FH University of Applied Sciences TECHNIKUM WIEN

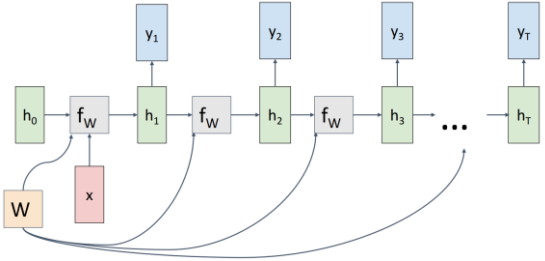
# RNNs: Unrolling

Same neural network structure is used for different sequential problems

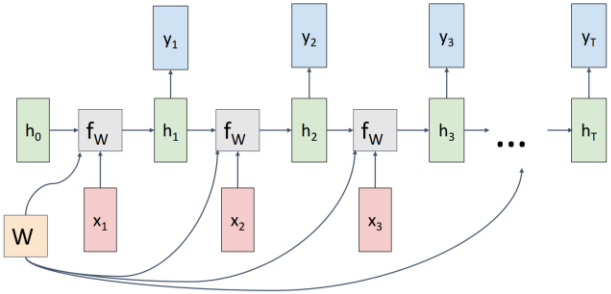
RNN Computational Graph (Many to One)




RNN Computational Graph (One to Many)




RNN Computational Graph (Many to Many)




**Always the same W matrices ☺**

# Backpropagation through time

Forward run outputs results:

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

$$\hat{y}_t = \sigma_o(W_{oh}h_t + b_o)$$

Total Loss function: $L = \sum_{t=1}^{T} L_t(\hat{y}_t, y_t)$

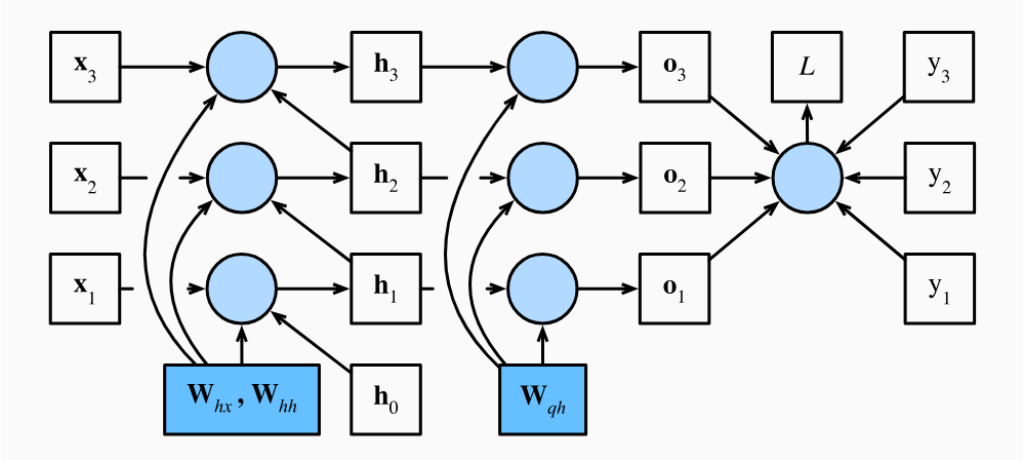
Backpropagation Process:

Update weights to minimize the loss

$$W_{xh} \leftarrow W_{xh} - \eta \cdot \frac{\partial L}{\partial W_{xh}}$$

$$W_{hh} \leftarrow W_{hh} - \eta \cdot \frac{\partial L}{\partial W_{hh}}$$

$$W_{oh} \leftarrow W_{oh} - \eta \cdot \frac{\partial L}{\partial W_{oh}}$$




https://d2l.ai/chapter_recurrent-neural-networks/bptt.html

**Backpropagation through time (BPTT):**

Unfold all the way to the initial step

(see last slide for an example)

$$h_t = f_W(h_{t-1}, x_t)$$

# RNNs: Pseudo-Code

Repeat till the stopping criterion is met:

1. Set all $h$ to zero.
2. Repeat for t = 0 to n-k
    1. Forward propagate the network over the unfolded network for $k$ time steps to compute all $h$ and $y$
    2. Compute the error as: $e = y_{t+k} - p_{t+k}$
    3. Backpropagate the error across the unfolded network and update the weights

FH University of Applied Sciences TECHNIKUM WIEN

# RNNs: Vanishing/exploding gradient problem

**Vanishing gradient:** if we keep multiplying with a weight smaller than 1.0 we will obtain a number very close to 0, taking only very small steps in our solution space. *Can lead to slow convergence or nonconvergence.*

**Exploding gradient:** if we keep multiplying with a weight larger than 1.0 we will obtain a number that is large. *Can lead to unstable convergence or divergence.*

| Method | Problem Addressed | Description |
|---|---|---|
| **Gated Architectures (LSTMs & GRUs)** | Vanishing gradients | Utilize gates to regulate information flow, facilitating gradient propagation over long sequences. |
| **Gradient Clipping** | Exploding gradients | Caps gradients when they exceed a certain threshold to prevent excessively large updates. |
| **Proper Activation Functions** | Vanishing gradients | Replacing sigmoid or tanh activations with ReLU or Leaky ReLU helps maintain gradient flow. |
| **Batch Normalization / Layer Normalization** | Both | Normalizes activations to stabilize training and improve gradient flow. |
| **Weight Initialization** | Both | Proper initialization prevents gradients from becoming too large or too small. |
| **Truncated Backpropagation Through Time (TBPTT)** | Vanishing gradients | Limits the number of steps in backpropagation, reducing long-term dependencies. |

# RNNs: Advantages and Disadvantages

**Advantages:**

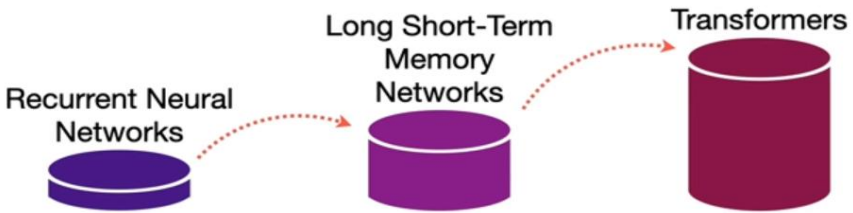
- Ability to handle sequential data and time series.
- Ability to handle inputs of varying lengths.
- Ability to store or ‘memorize’ temporal information.

**Disadvantages:**

- Slow training: requires high processing.
- Erratic gradients: vanishing or exploding gradient problem. This can make training difficult and time-consuming.

# Other RNN Architectures

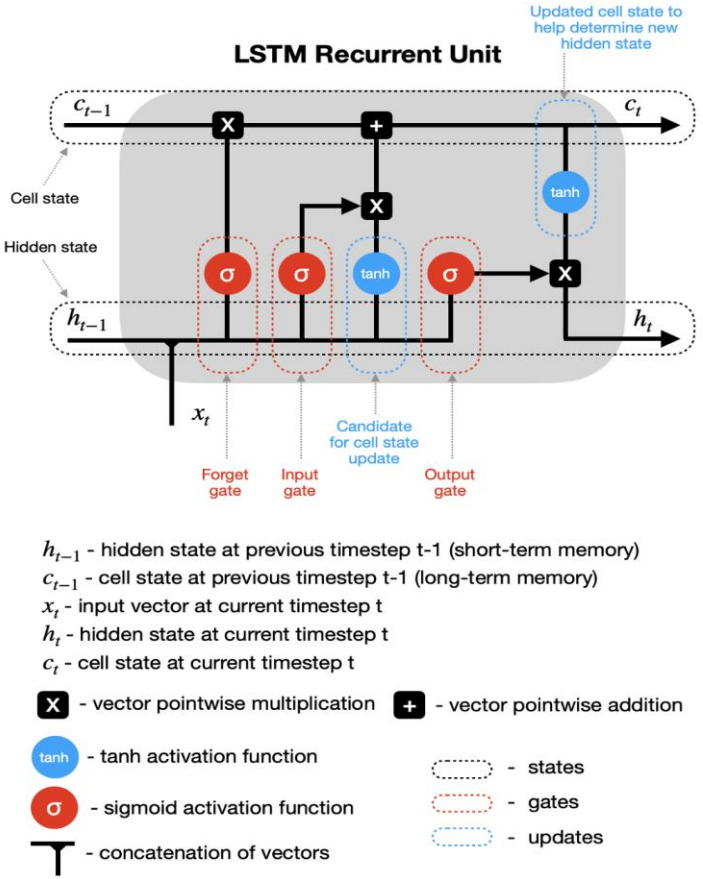
- RNNs by themselves are not well performant due to the vanishing/exploding gradient problem
- RNNs have been refined to other architectures in order to overcome these limitations
  (see Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications)
- In certain applications (e.g. natural language processing) Transformers have virtually replaced RNNs/LSTMs



# Other RNN Architectures

**Long Short Term Memory (LSTM):** networks designed to address the vanishing gradient problem in RNNs. LSTMs use three gates (called **input, output, and forget gate**) to pass information between long term and short term memory cells.
See this video for a detailed explanation.

**Gated Recurrent Units (GRU):** also designed to handle the vanishing gradient problem through gates. They use **reset and update gates** to determine which information is to be retained for future predictions, similarly to LSTMs.

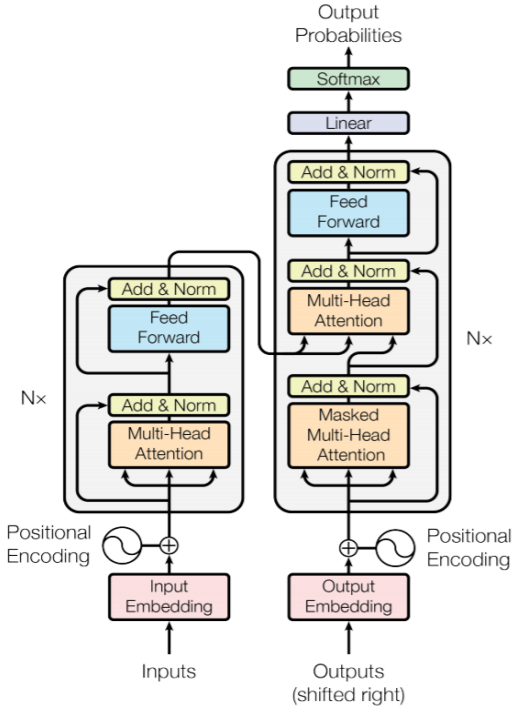



FH University of Applied Sciences TECHNIKUM WIEN

# Other RNN Architectures

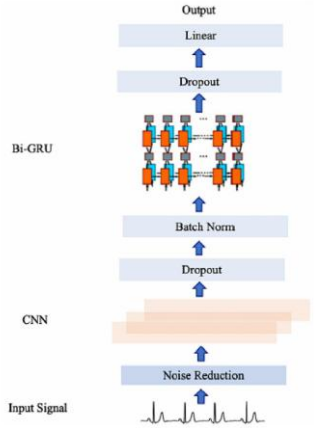
**Bidirectional Recurrent Neural Networks (BRNN):** inputs from future time steps are used to improve the accuracy of the network. It is like knowing the first and last words of a sentence to predict the middle words.

**Transformers:** use self attention mechanisms to provide context. Determining the significance of each part of the input sequence relative to others allows transformer to capture long term dependencies.
See paper walkthrough in [this video](#).



# RNN Applications

## Signal Data

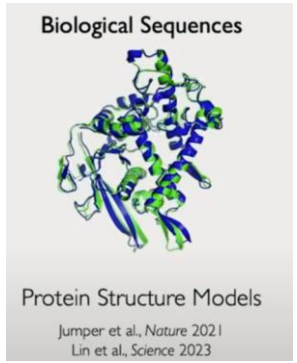



www.nature.com/scientificreports

scientific reports

OPEN **Deep residual-dense network based on bidirectional recurrent neural network for atrial fibrillation detection**

Asif Ali Laghari[1], Yanqiu Sun[2✉], Musaed Alhussein[4], Khursheed Aurangzeb[4], Muhammad Shahid Anwar[5] & Mamoon Rashid[3]




International Journal of Engineering and Innovative Research 6:1 (2024) 15-28

International Journal of Engineering and Innovative Research

http://dergipark.gov.tr/ijeir

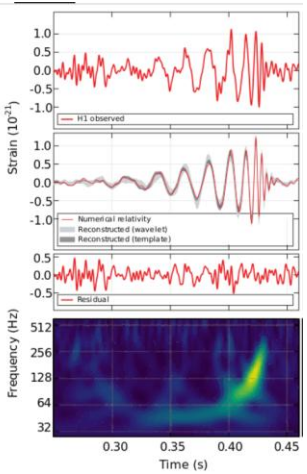
**RNN-Based Time Series Analysis for Wind Turbine Energy Forecasting**

Selahattin Barış Çelebi [1], Şehmus Fidan[2*].

Open Access Article

**Earthquake Detection Using Stacked Normalized Recurrent Neural Network (SNRNN)**

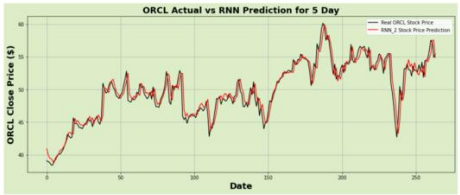
by Muhammad Atif Bilal [1,*], Yongzhi Wang [2,3,*], Yanju Ji [1], Muhammad Pervez Akhter [4] and Hengxi Liu [2]




Open Access Article

**Comparative Analysis of Recurrent Neural Networks in Stock Price Prediction for Different Frequency Domains**

by Polash Dey [1,†], Emam Hossain [1,†], Md. Ishtiaque Hossain [2,†], Mohammed Armanuzzaman Chowdhury [2], Md. Shariful Alam [3], Mohammad Shahadat Hossain [2] and Karl Andersson [4,*]




Conferences > 2021 IEEE 32nd International ...

**Accelerating Recurrent Neural Networks for Gravitational Wave Experiments**

Publisher: IEEE Cite This PDF

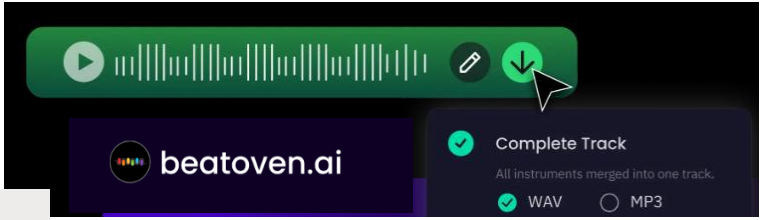
Zhiqiang Que ; Erwei Wang ; Umar Marikar ; Eric Moreno ; Jennifer Ngadiuba ; Hamza Javed All Authors

FH University of Applied Sciences TECHNIKUM WIEN

# RNN Applications

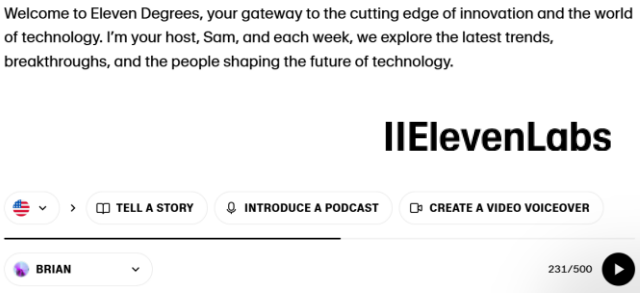
## Generative Models





A video generated using OpenAI's Sora text-to-video model, using the prompt: `A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the colorful lights. Many pedestrians walk about.`


FH University of Applied Sciences TECHNIKUM WIEN

# Assignment: RNNs

a) Explain recurrent neural networks using 3 slide.
   Use self-made images or even hand drawings (of which you take a photo).
   Use self-written explanations.
   Do not copy from the lecture slides or the internet (neither text nor images)

b) Choose an advanced method from RNNs to explain in terms of a pseudo code:
   LSTMs, GRU, Bi-directional RNNs, Transformers.

Provide a short 1 slide explanation of the method.

# RNNs: Example

Weekend plans based on the weather:

**Museum**

**Movies**


**Party**

**Sunny**

**Rainy**

Follows same sequence: park-movies-party.
If it is sunny, does the same as previous week.

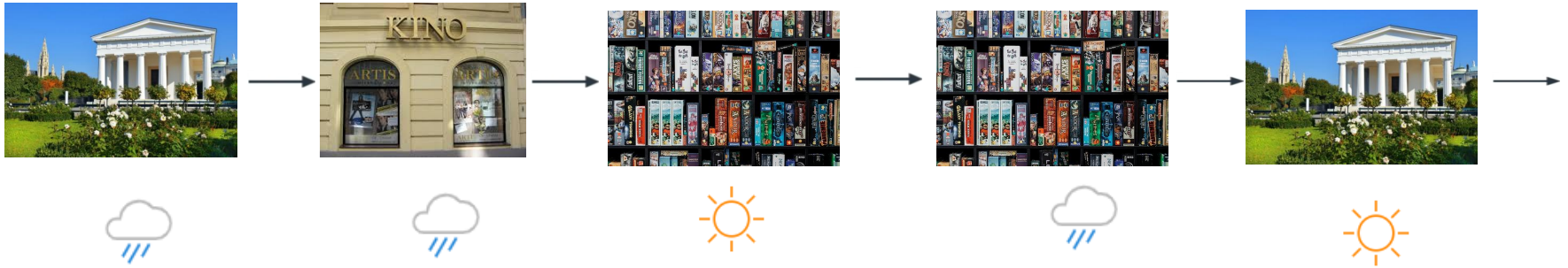
Check an equivalent example at

[A friendly introduction to Recurrent Neural Networks]

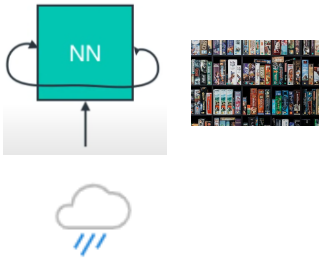
# RNNs: Example

Weekend plans based on the weather forecast:

1st Weekend 2nd Weekend 3rd Weekend 4th Weekend 5th Weekend

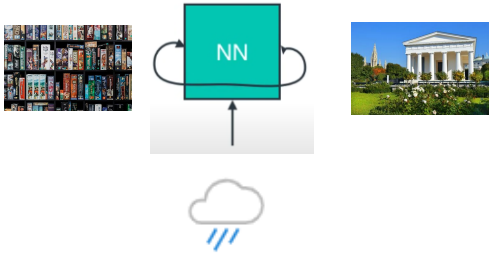



If the forecast is rain and went last week to a party




Output fed as input in t+1

Predicts going to the museum



# RNNs: Example

Vector Representation

**Museum**

$\begin{bmatrix}1\\0\\0\end{bmatrix}$ 

**Movies**

$\begin{bmatrix}0\\1\\0\end{bmatrix}$ 


**Party**

$\begin{bmatrix}0\\0\\1\end{bmatrix}$ 

**Activity Matrix:** concatenates the same vector to the next one

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\\hline 0&0&1\\1&0&0\\0&1&0\end{bmatrix} \begin{bmatrix}0\\1\\0\end{bmatrix}$$ 

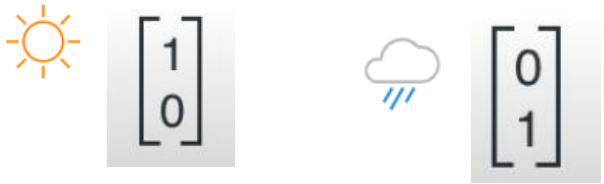

$$= \begin{bmatrix}0\\1\\0\\\hline 0\\0\\1\end{bmatrix}$$


Same

Next activity

# RNNs: Example

Vector Representation:




**Weather Matrix:** concatenates the same vector to the next week


# RNNs: Example

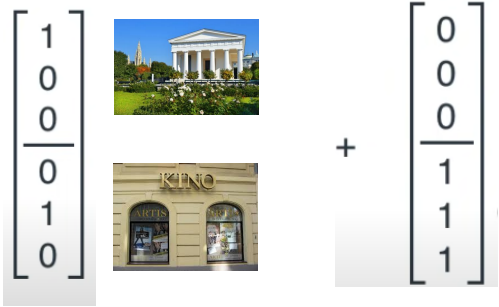
Activity

Weather

Initial Case

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

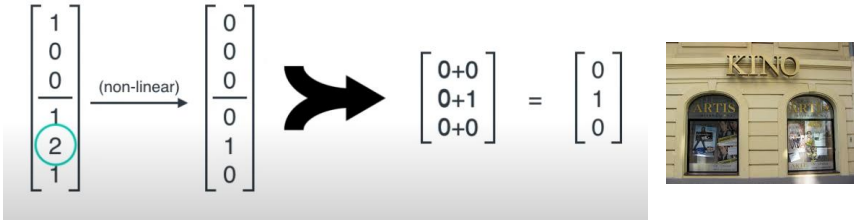
$$\hat{y}_t = \sigma_o(W_{oh}h_t + b_o)$$




Same

Next week

Activation function



# BPTT: Example for 3 steps

## A. Forward Pass for steps 1, 2, and 3

$$h_1 = \sigma_h(W_{xh}x_1 + W_{hh}h_0 + b_h)$$

$$\hat{y}_1 = \sigma_o(W_{oh}h_1 + b_o)$$

$$h_2 = \sigma_h(W_{xh}x_2 + W_{hh}h_1 + b_h)$$

$$\hat{y}_2 = \sigma_o(W_{oh}h_2 + b_o)$$

$$h_3 = \sigma_h(W_{xh}x_3 + W_{hh}h_2 + b_h)$$

$$\hat{y}_3 = \sigma_o(W_{oh}h_3 + b_o)$$

$$L = \sum_{t=1}^{3} L_t(\hat{y}_t, y_t)$$

## B. Backward Pass for steps 3, 2 and 1

### Output layer

$$\delta_3 = \frac{\partial L_3}{\partial \hat{y}_3} \cdot \sigma_o'(z_3)$$

$$\left.\frac{\partial L}{\partial W_{oh}}\right|_{t=3} = \delta_3 \cdot h_3^T$$

$$\delta_2 = \frac{\partial L_2}{\partial \hat{y}_2} \cdot \sigma_o'(z_2)$$

$$\left.\frac{\partial L}{\partial W_{oh}}\right|_{t=2} = \delta_2 \cdot h_2^T$$

$$\delta_1 = \frac{\partial L_1}{\partial \hat{y}_1} \cdot \sigma_o'(z_1)$$

$$\left.\frac{\partial L}{\partial W_{oh}}\right|_{t=1} = \delta_1 \cdot h_1^T$$

### Hidden Layer

$$\frac{\partial L}{\partial h_3} = \delta_3 \cdot W_{oh}^T$$

$$\frac{\partial L}{\partial h_2} = \delta_2 \cdot W_{oh}^T + \left(\frac{\partial L}{\partial h_3} \cdot \sigma_h'(a_3) \cdot W_{hh}^T\right)$$

$$\frac{\partial L}{\partial h_1} = \delta_1 \cdot W_{oh}^T + \left(\frac{\partial L}{\partial h_2} \cdot \sigma_h'(a_2) \cdot W_{hh}^T\right)$$

FH University of Applied Sciences TECHNIKUM WIEN

# BPTT: Example for 3 steps

## C. Accumulated Gradients

**Output layer**

$$\frac{\partial L}{\partial W_{oh}} = \sum_{t=1}^{3} \delta_t \cdot h_t^T$$

**Hidden Layer**

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{3} \left( \frac{\partial L}{\partial h_t} \cdot \sigma_h'(a_t) \cdot h_{t-1}^T \right)$$

**Input Layer**

$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^{3} \left( \frac{\partial L}{\partial h_t} \cdot \sigma_h'(a_t) \cdot x_t^T \right)$$

## D. Updated Weights

**Input Layer**

$$W_{xh} \leftarrow W_{xh} - \eta \cdot \frac{\partial L}{\partial W_{xh}}$$

**Hidden Layer**

$$W_{hh} \leftarrow W_{hh} - \eta \cdot \frac{\partial L}{\partial W_{hh}}$$

**Output layer**

$$W_{oh} \leftarrow W_{oh} - \eta \cdot \frac{\partial L}{\partial W_{oh}}$$

FH University of Applied Sciences TECHNIKUM WIEN