Machine Learning in Biomedical Sciences and Bioengineering

# Lecture 8
# Recurrent Neural Networks (RNN)

2025 version 1.00

James Choi

**Part 1. Recurrent Neural Networks**

**Part 2. Live Coding Demonstration**

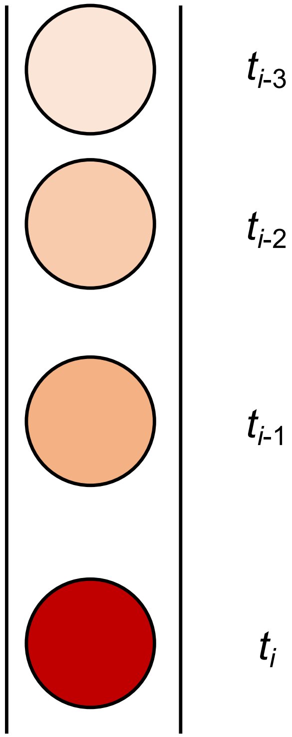
# Limitations with basic neural networks

- Basic Neural Networks and Convolutional Neural Networks (CNNs) are constrained:
  - **Inputs** are a **fixed-sized vector**
  - **Outputs** are a **fixed-sized vector**

- The neural networks operate with a **fixed number of computational steps.**
  - For example, the number of layers are fixed

# Sequence modelling

- Build **neural networks** that can handle and learn from sequential data.




The tumour is benign.

The tumour is not benign.

GCTGGCATACTTTTACGGCTA

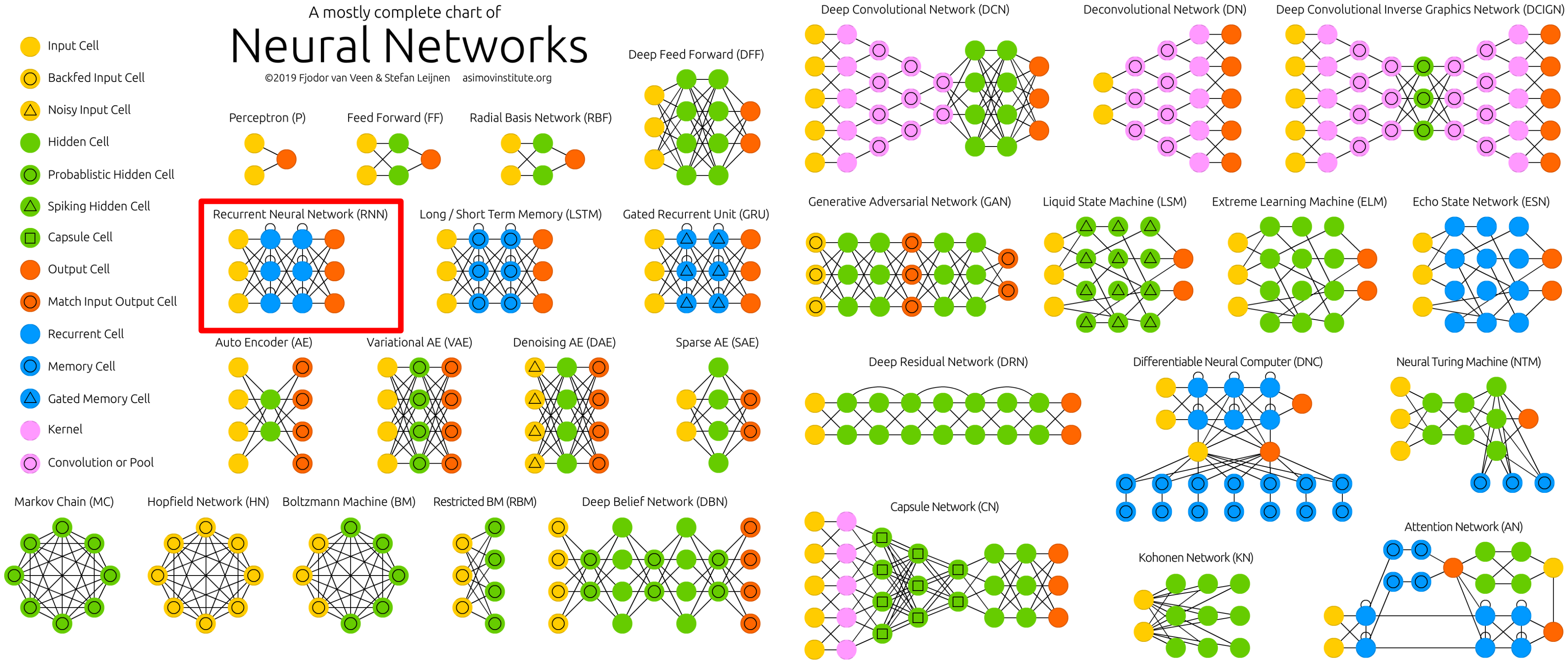
**The order of the information is critical to the meaning of the data.**

# Recurrent neural networks (RNN)

- **Recurrent neural networks (RNN)** are a specialised neural network architecture characterized by a bi-directional flow of information between its layers.

- In contrast to uni-directional feedforward neural networks, an RNN allows the output from some nodes to affect subsequent input to the same nodes.

# Neural networks

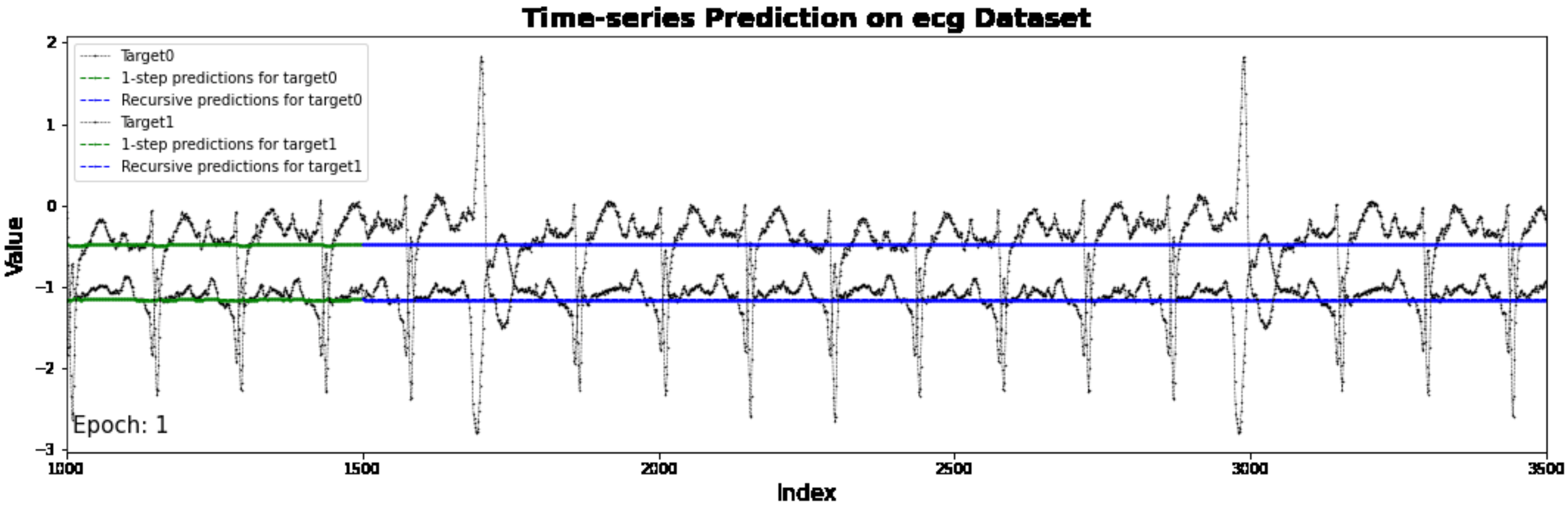



https://www.asimovinstitute.org/neural-network-zoo/

# Examples

- Acoustic signals (e.g., audio, ultrasound)
- Electrical signals (e.g., ECG)
- DNA and RNA sequences
- Protein sequences

# ECG



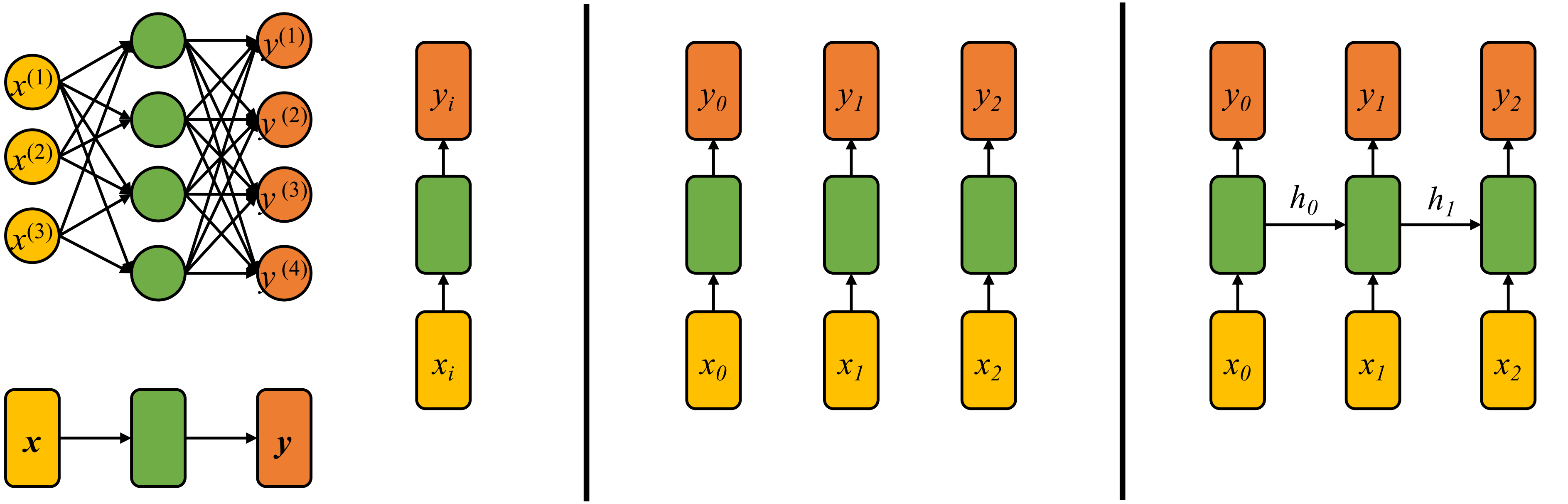

https://github.com/immanuvelprathap/Electrocardiogram-Anomaly-Detection-RNN-Time-Series

# Recurrent Neural Networks


Feed-forward networks

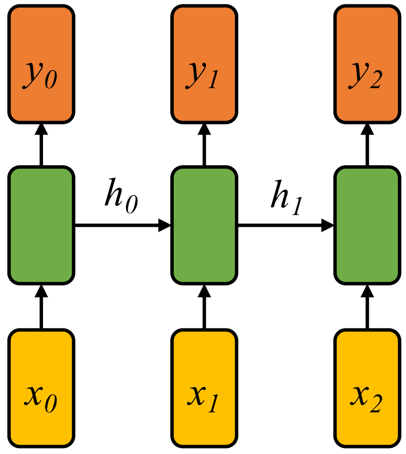
Independently evaluated models

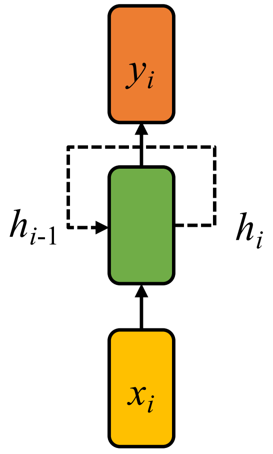
Recurrent Neural Network



# Recurrent Neural Networks

Recurrent Neural Network

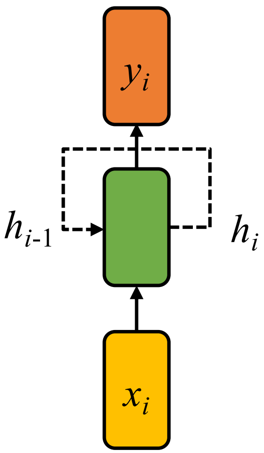


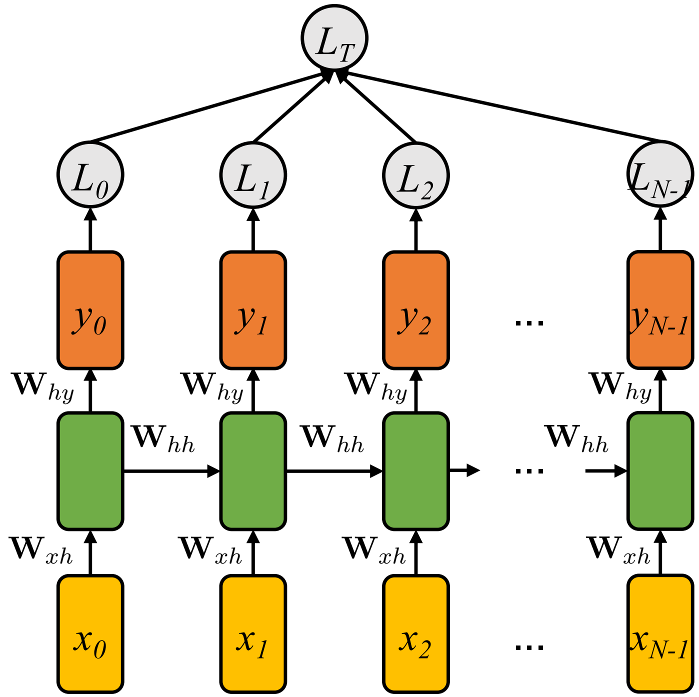




$$y_i = f(x_i, h_{i-1})$$

$$h_i = f_W(x_i, h_{i-1})$$

# Recurrent Neural Networks




$$y_i = f(x_i, h_{i-1})$$

$$h_i = f_W(x_i, h_{i-1})$$

$$h_i = tanh(\mathbf{W}_{hh}^T h_{i-1} + \mathbf{W}_{xh}^T x_i)$$

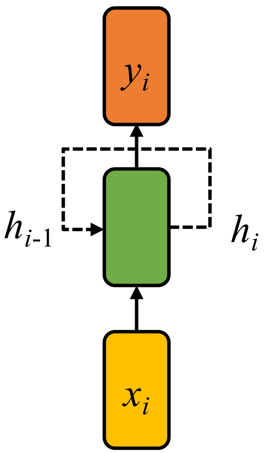
$$\hat{y}_i = \mathbf{W}_{hy}^T h_i$$

The **loss** is calculated for every output and then **summed**

The **same weight matrices** are used at every sequence step.

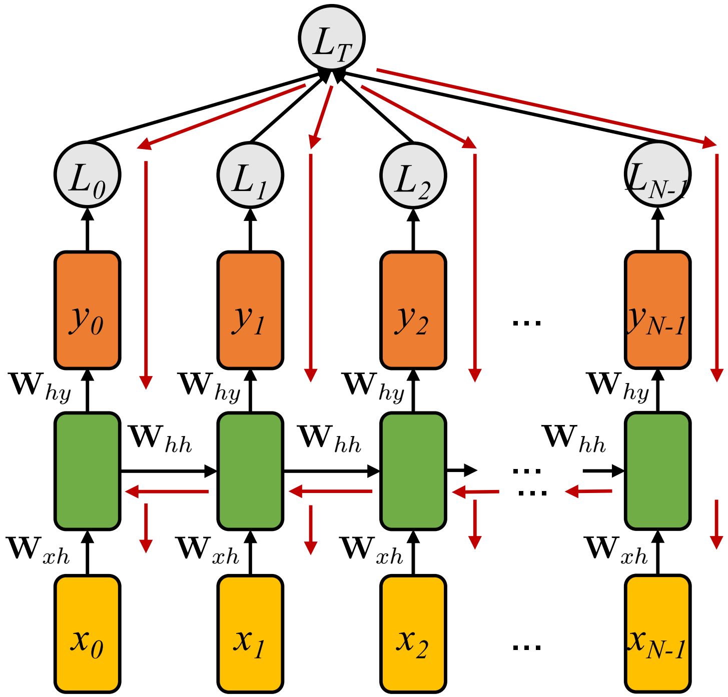
$\mathbf{W}_{xh}$ $\mathbf{W}_{hh}$ $\mathbf{W}_{hy}$

# Recurrent Neural Networks




$$y_i = f(x_i, h_{i-1})$$

$$h_i = f_W(x_i, h_{i-1})$$




$$h_i = tanh(\mathbf{W}_{hh}^T h_{i-1} + \mathbf{W}_{xh}^T x_i)$$

$$\hat{y}_i = \mathbf{W}_{hy}^T h_i$$

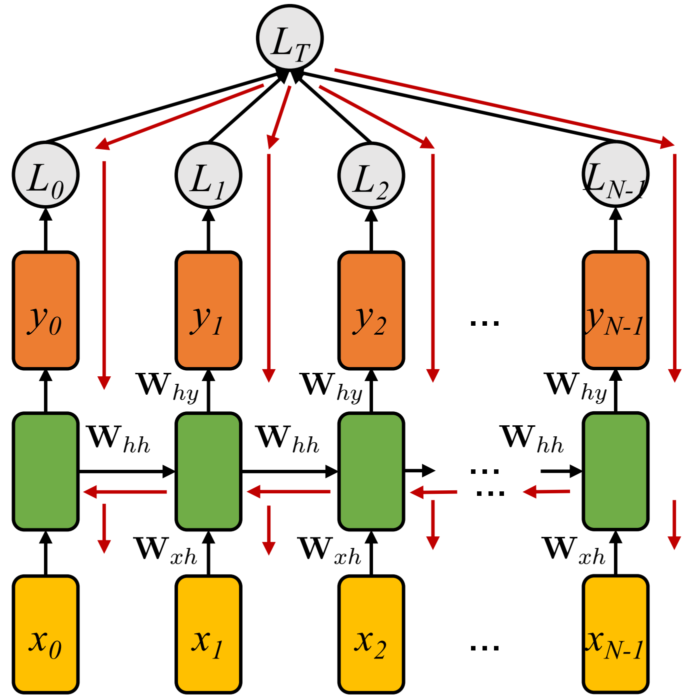
Need to backpropagate to find the **gradient for all the weights**

Propagating through the hidden states is complex, computationally expensive, and can lead to peculiar (and poor) results.

# Back propagation concerns

- Many of the values are >1
  - Problem: **Exploding gradients**
  - Solution:
    - Gradient clipping

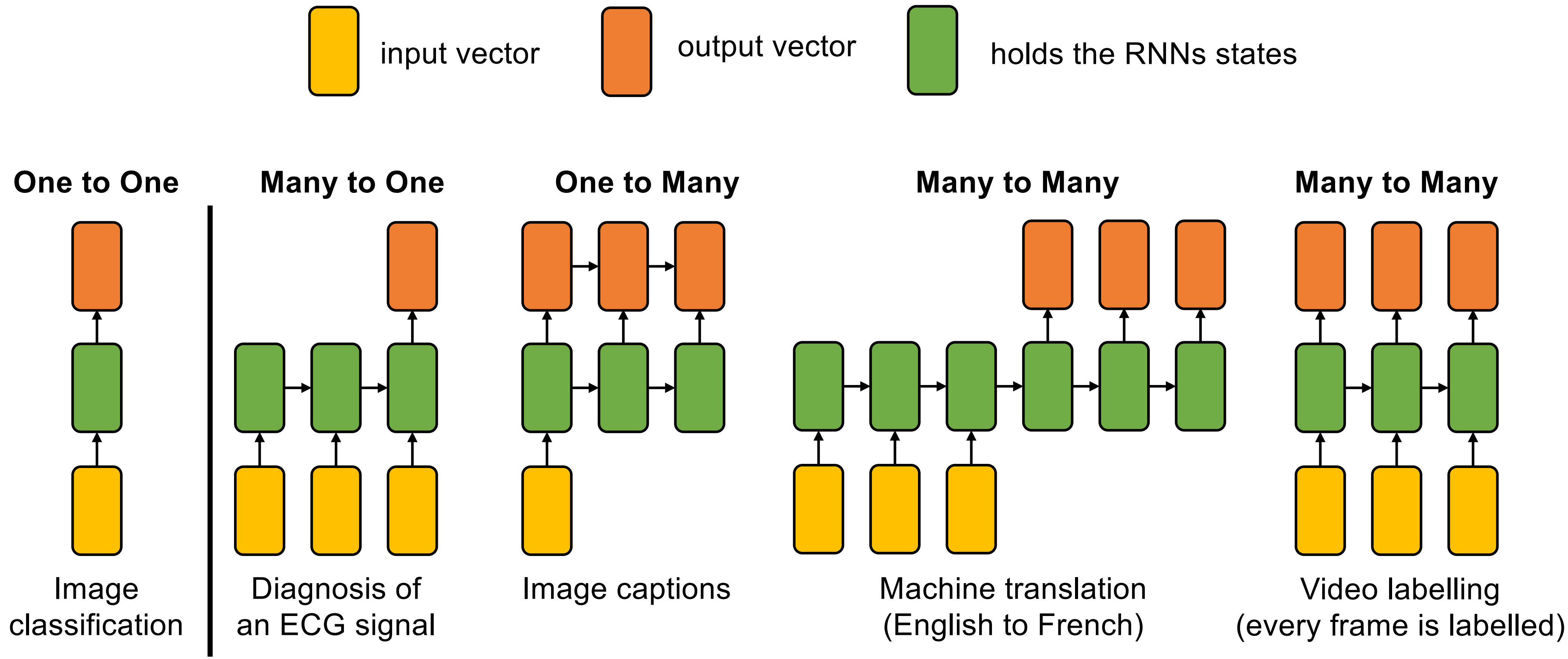
- Many of the values are < 1
  - Problem: **Vanishing gradients**
  - Solutions:
    - Activation function
    - Weight initialisation.
      - Initialise to identity matrix.
    - Network architecture
      - LSTM




$$h_i = tanh(\mathbf{W}_{hh}^T h_{i-1} + \mathbf{W}_{xh}^T x_i)$$

$$\hat{y}_i = \mathbf{W}_{hy}^T h_i$$

# Types of RNNs




**There is no constraint to the number of inputs, hidden layers, or outputs.**

Adapted from https://karpathy.github.io/2015/05/21/rnn-effectiveness/

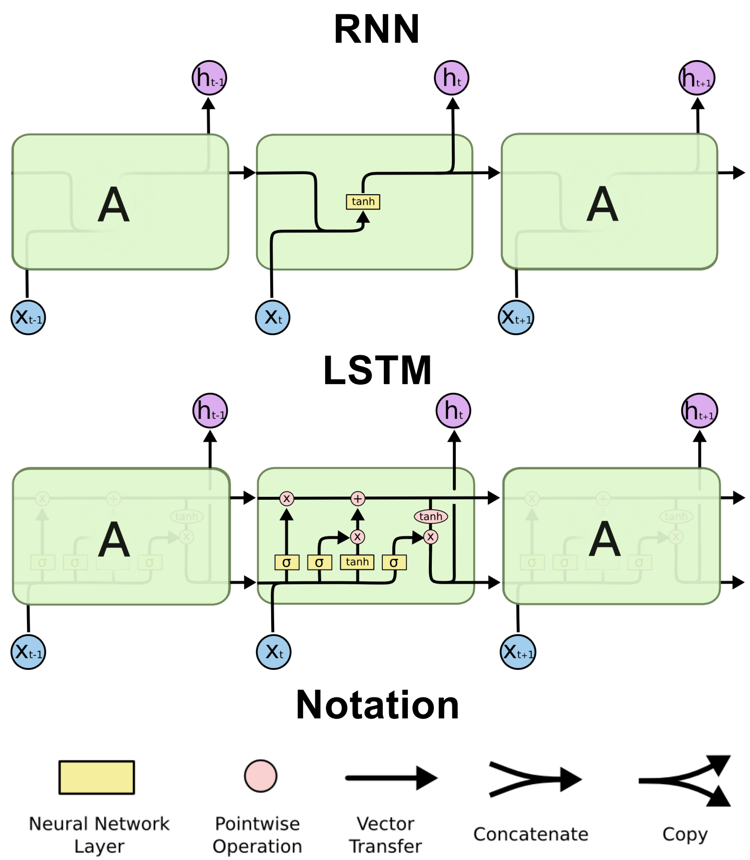
# Limitations with RNNs

- Vanishing and exploding gradients.
  - During backpropagation, gradients can shrink or grow exponentially.
  - This makes learning long-range dependencies difficult (vanishing) or unstable (exploding.
  - Them model may forget earlier parts of the sequence or fail to converge
- Difficulty learning long-term dependencies
- Sequential processing bottleneck
  - RNNs process inputs one step at a time.
  - Training and inference cannot be easily parallelised across steps, leading to slower computations than more recent models (eg Transformers)
- Limited memory capacity
  - The hidden state is often a bottleneck for storing complex or long sequences of information
  - The model may compress or discard important contextual information.
- Bias toward recent inputs
  - Due to gradient decay, RNNs tend to prioritise more recent inputs over older ones.
- Training instability
  - Sensitive to weight initialization, learning rate, and sequence length
- Less interpretability
  - Hidden state dynamics are hard to analyse or visualise

# Long short term memory (LSTM)

- **Long short term memory (LSTM)** is a type of **recurrent neural network (RNN)**.

- LSTM can control the information flow:
  - **Forget**. Get rid of irrelevant information
  - **Store**. Store relevant information
  - **Update**. Selectively update cells.
  - **Output**. Return a filtered version of the cell state.




https://colah.github.io/posts/2015-08-Understanding-LSTMs/

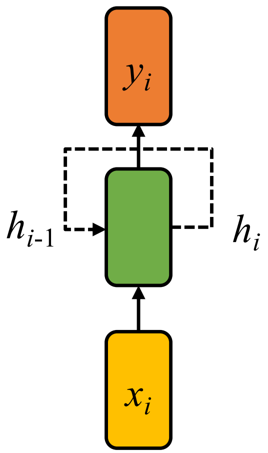
**Part 1. Recurrent Neural Networks**

**Part 2. Live Coding Demonstration**

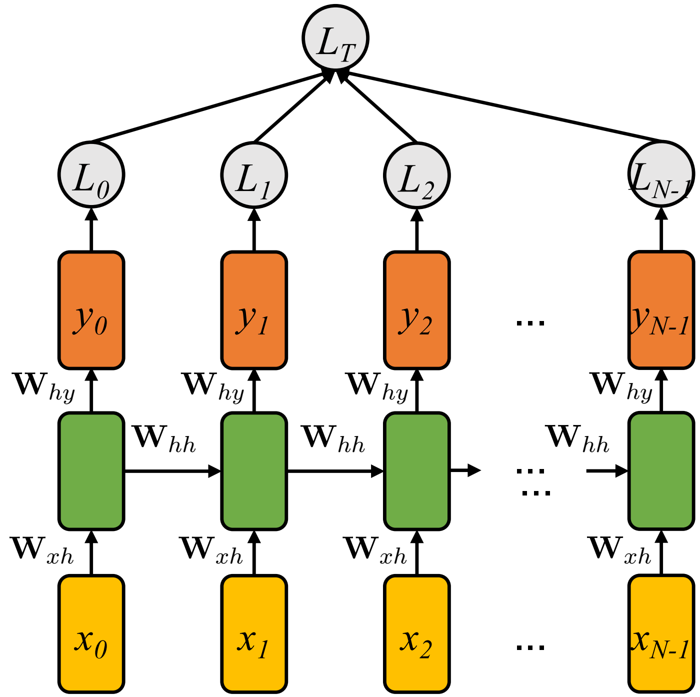
# Live coding demonstration

- Demo was adapted from a PyTorch Tutorial:
  - https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html

# Recurrent Neural Networks




$$y_i = f(x_i, h_{i-1})$$
$$h_i = f_W(x_i, h_{i-1})$$




$$h_i = tanh(\mathbf{W}_{hh}^T h_{i-1} + \mathbf{W}_{xh}^T x_i)$$
$$\hat{y}_i = \mathbf{W}_{hy}^T h_i$$