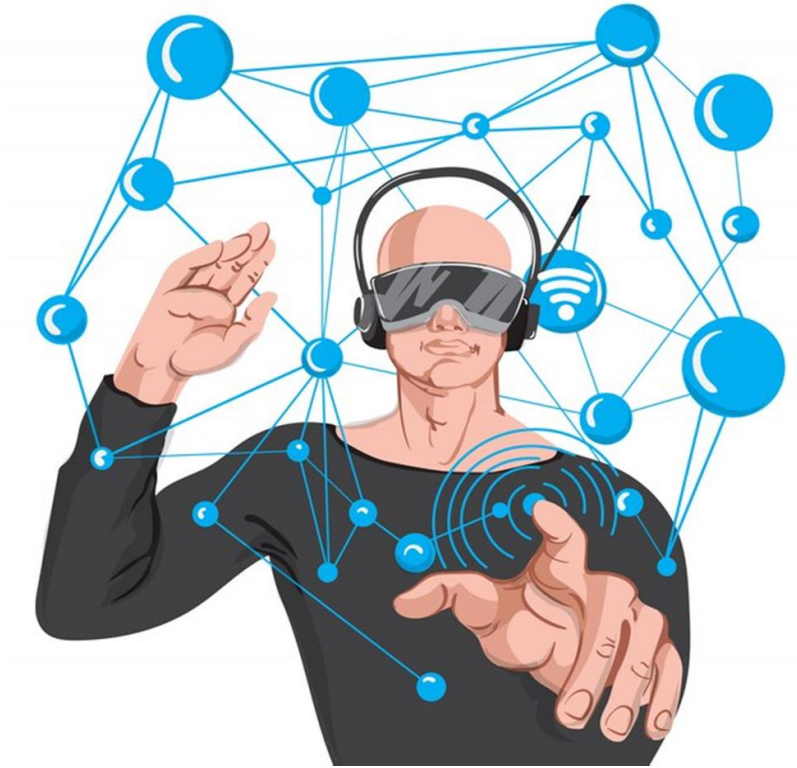
Building Large Language Model Applications

Neural Network Models in NLP

Hamza Farooq
Dr. Saima Hassan

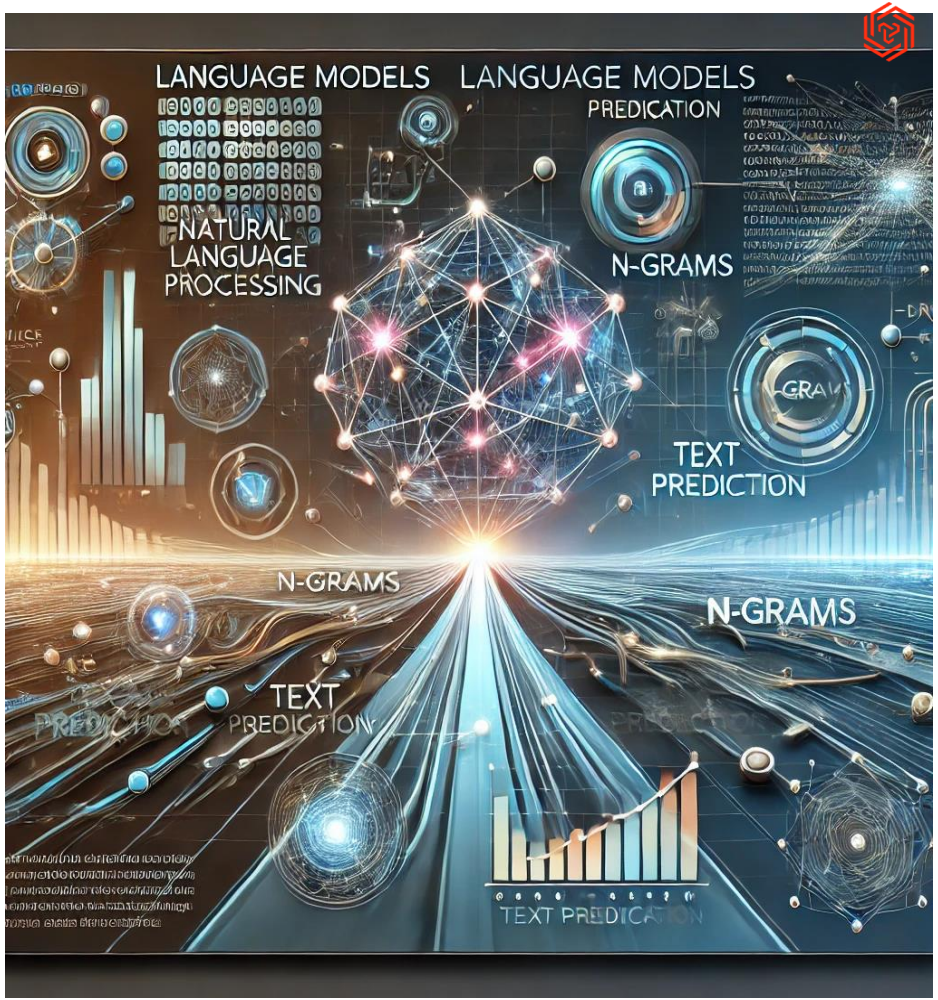
Learning outcomes

- What are Neural Networks?
- What is Recurrent NN?
- LSTMs
- GRUs



Language models

Language models are computational systems capable of **understanding** and **generating text in natural language**. They possess the transformative ability **to predict the likelihood** of a given sequence of words or generate new text based on specified input data. The most common type of language models are those based on **N-grams**. These models estimate word **probabilities based on context** to predict the next element in a sequence.



Why we move Neural Network Modeling

Challenges with Traditional Models:

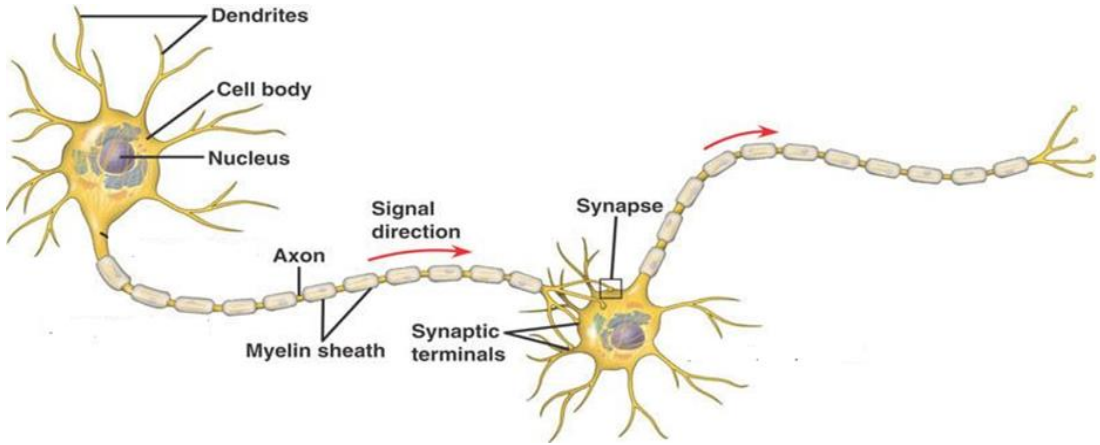
- Curse of dimensionality in N-grams
- Lack of context awareness beyond fixed windows
- Difficulty capturing complex dependencies between words

Why Neural Networks?

- Learn hierarchical representations of data
- Capture long-term dependencies
- Improve prediction accuracy

Basic Components of Biological Neurons

Neuron is the elementary nerve cell that process information.



Dendrites: Tree-like branches, responsible for receiving the information from other neurons

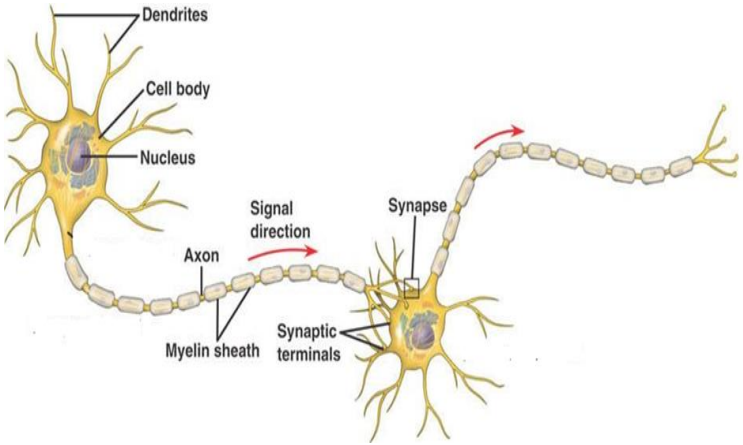
Soma: Cell body of the neuron and is responsible for processing of information, received from dendrites.

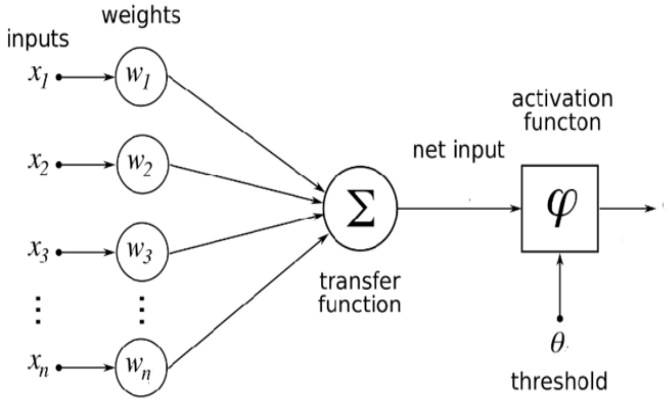
Axon: It is fiber acting like transmission lines that sends information.

Synapses : It is the connection between the axon and other neuron dendrites.

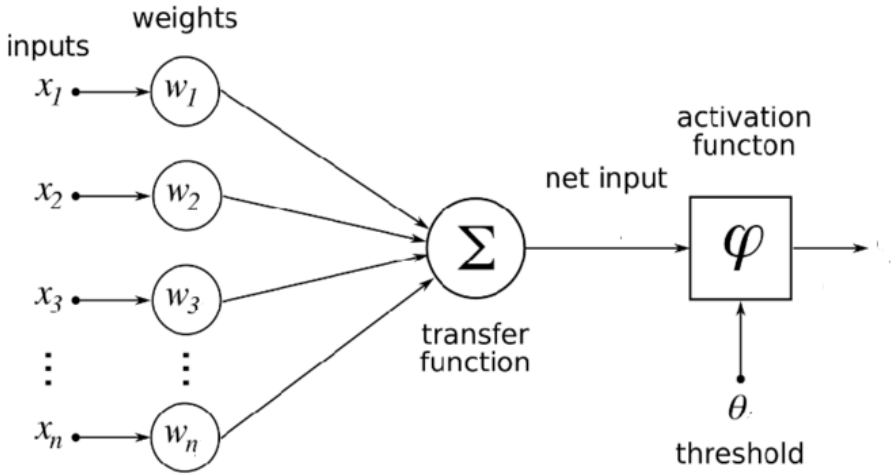
So, what are Artificial Neurons?

Neural Networks are similar to our own brains; they have ‘neurons’ and a network architecture (like our brain), and are great at learning from a great variety of examples.





Artificial Neuron



The first computational machine of a neuron was proposed by Warren McCulloch and Walter Pitts in 1943.

$$y = x_1 . w_1 + x_2 . w_2 + x_3 . w_3 \ldots x_n . w_n$$

i.e., Net input $y = \sum_i^n x_i . w_i$

Threshold Logic Unit

Definition and Examples

Definition A **threshold logic unit** is a simple processing unit for real-valued numbers with n inputs $x_1, \ldots, x_n$ and one output y. The unit as a whole possesses a **threshold** θ. To each input x_i a **weight** w_i is assigned. A threshold logic unit computes the function

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0 & \text{otherwise.} \end{cases}$$

The inputs are often combined into an input vector $\mathbf{x} = (x_1, \ldots, x_n)$ and the weights into a weight vector $\mathbf{w} = (w_1, \ldots, w_n)$. With the help of the scalar product, the condition tested by a threshold logic unit may then also be written as $\mathbf{wx} \geq \theta$.

Example

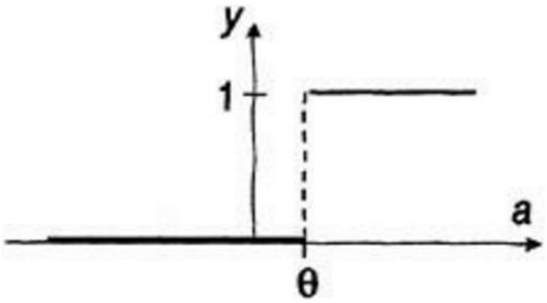


Figure 2.3 Activation-output threshold relation in graphical form.

As an example, consider a five-input unit with weights (0.5, 1.0, -1.0, -0.5, 1.2), that is w_1=0.5, w_2=1.0,…, w_5=1.2, and suppose this is presented with inputs (1, 1, 1, 0, 0) so that x_1=1, x_2=1,…, x_5=0. suppose that **θ** =0.2

u = (0.5 x 1) + (1.0 x 1) + (−1.0 x 1) + (−0.5 x 0) + (1.2 x 0)
u = 0.5

y =f(u) = f(0.5)
y = 1

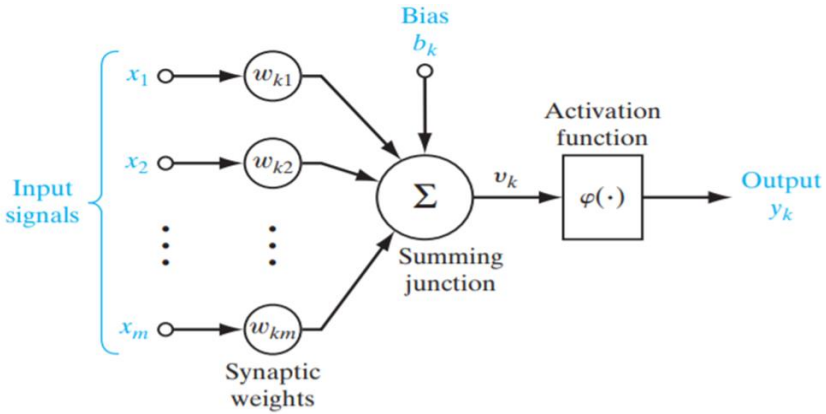
There are however several limitations to McCulloch-Pitts Neurons :

- It cannot process non-boolean inputs
- It gives equal weights to each input
- The threshold θ much be chosen by hand

For all these reasons, a necessary upgrade was required.

The Rosenblatt's Neuron

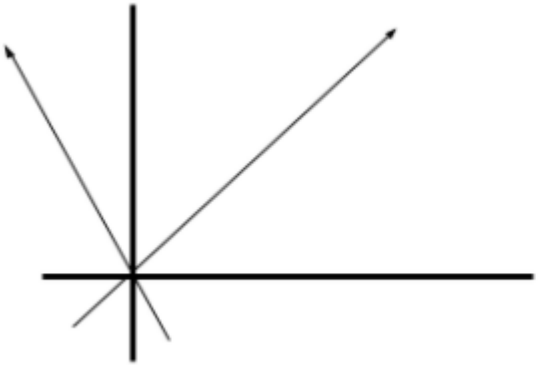
The Rosenblatt's Neuron

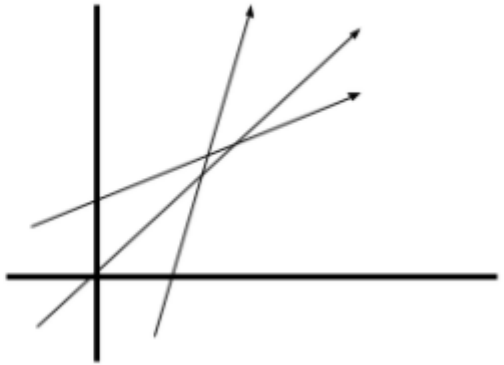


$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \varphi(u_k + b_k)$$

Effects of “bias” in Artificial Neuron





Due to absence of bias, the model will train over a point passing through origin only, which is not in accordance with real-world scenario. Also, with the introduction of bias, the model will become more flexible.

The bias neuron makes it possible to move the activation function left, right, up, or down on the number graph.

https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/

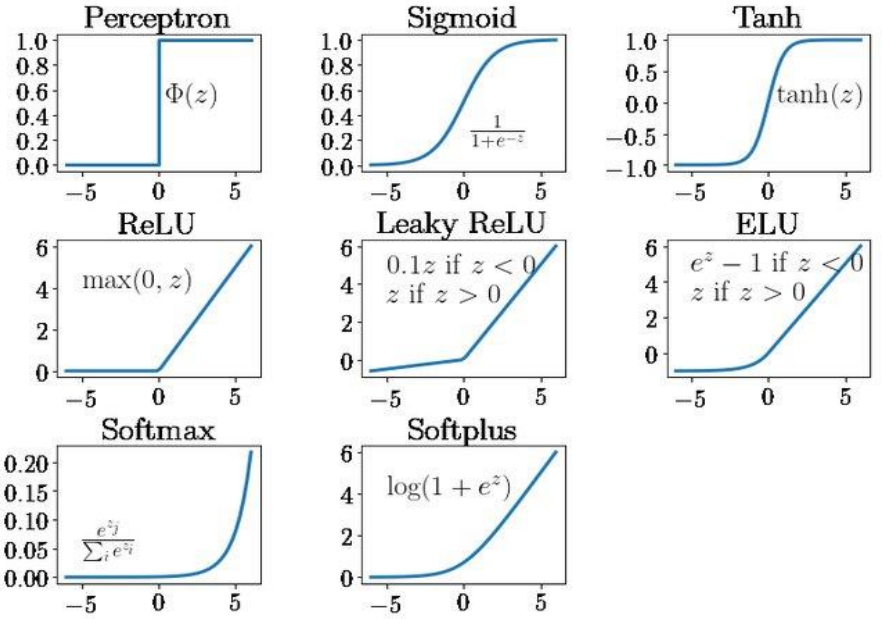
Activation Function

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

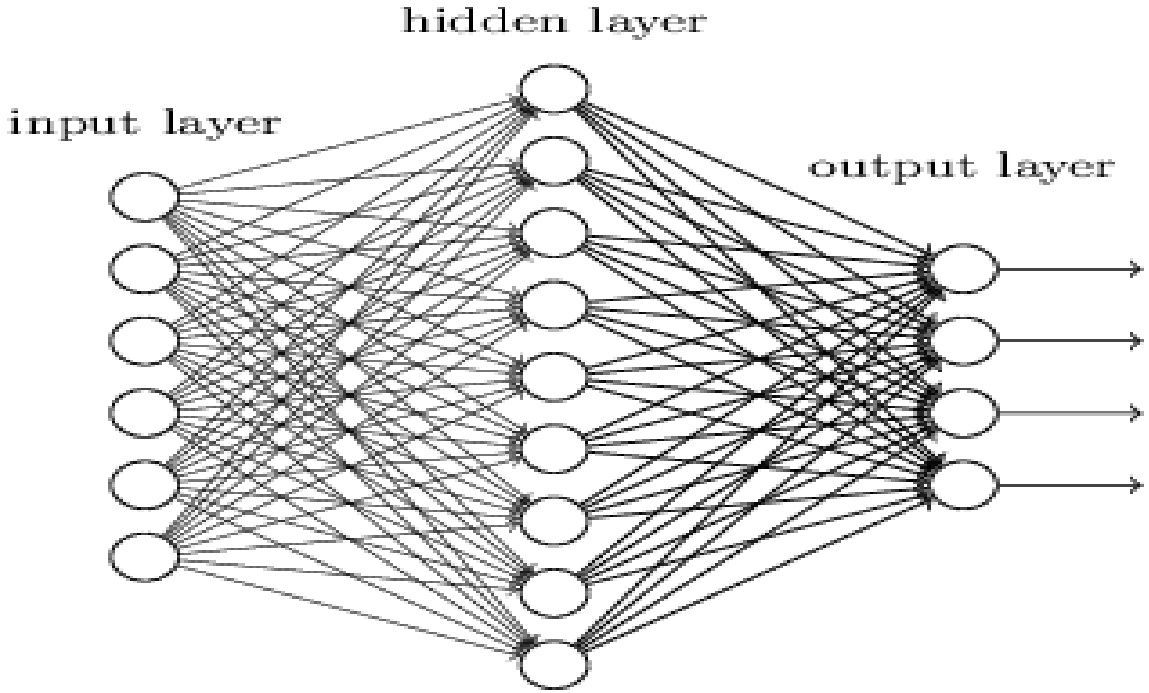
Activation Function

- **Sigmoid:** Outputs a value between 0 and 1, used in binary classification.
- **Tanh:** Outputs values between -1 and 1, used for normalizing data.
- **ReLU:** Sets all negative values to zero, widely used for deep learning.
- **Leaky ReLU:** Allows a small, non-zero slope for negative values, addressing ReLU's "dying" problem.



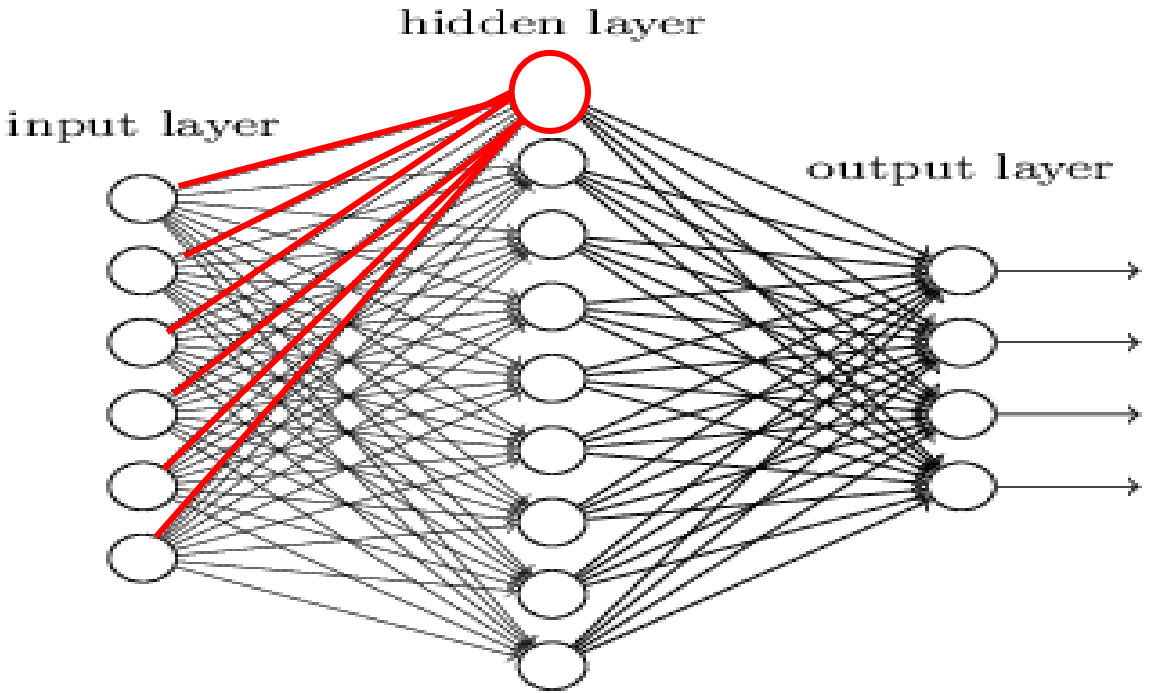
DOI: 10.48550/arXiv.2005.05235

Multilayer Artificial Neural Network



A **Multilayer ANN** is a powerful neural network with multiple layers that allow it to **extract features**, **recognize patterns**, and **learn from data**, making it essential for **deep learning** tasks.

Multilayer Artificial Neural Network



Book Suggestion: **An Introduction to Neural Networks for Beginners** by **Dr Andy Thomas**

Training an Artificial Neural Network

1. **Initialize the weights** to 0 or small random numbers.

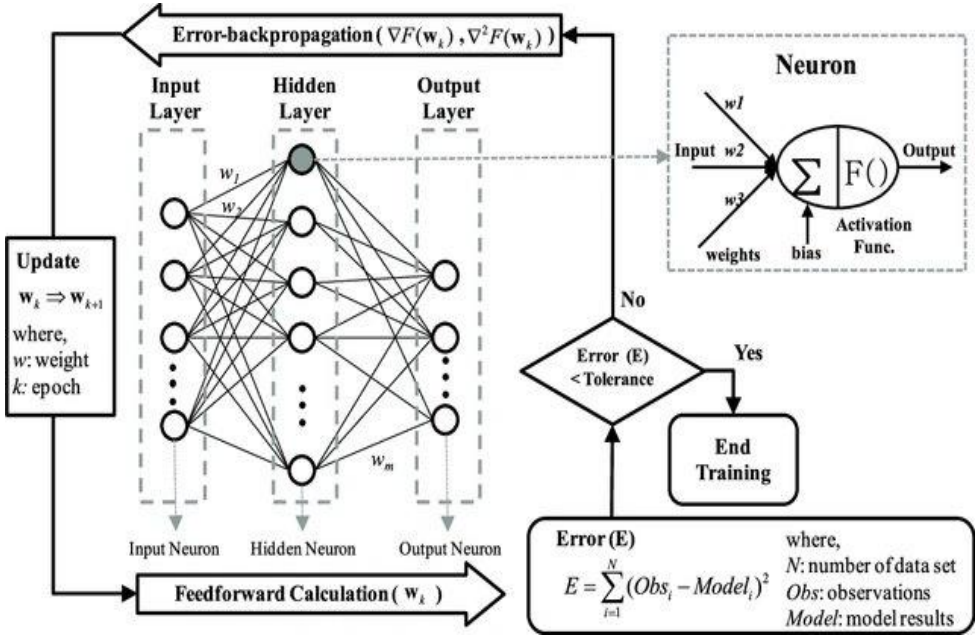
2. **Calculate the output**

   Forward Propagation: Input data flows through the network to calculate output.

1. **Loss Calculation:** Compares predicted output with actual output using a loss function.

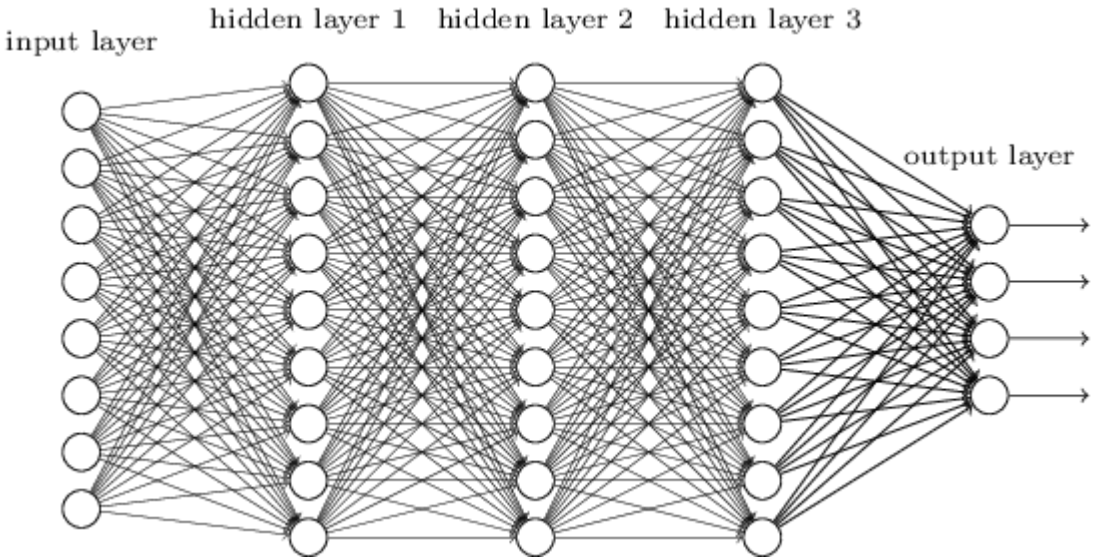
2. **Update the weights**

   Backpropagation: Error is propagated back through the network to adjust weights.



DOI: 10.1016/j.jher.2014.09.006

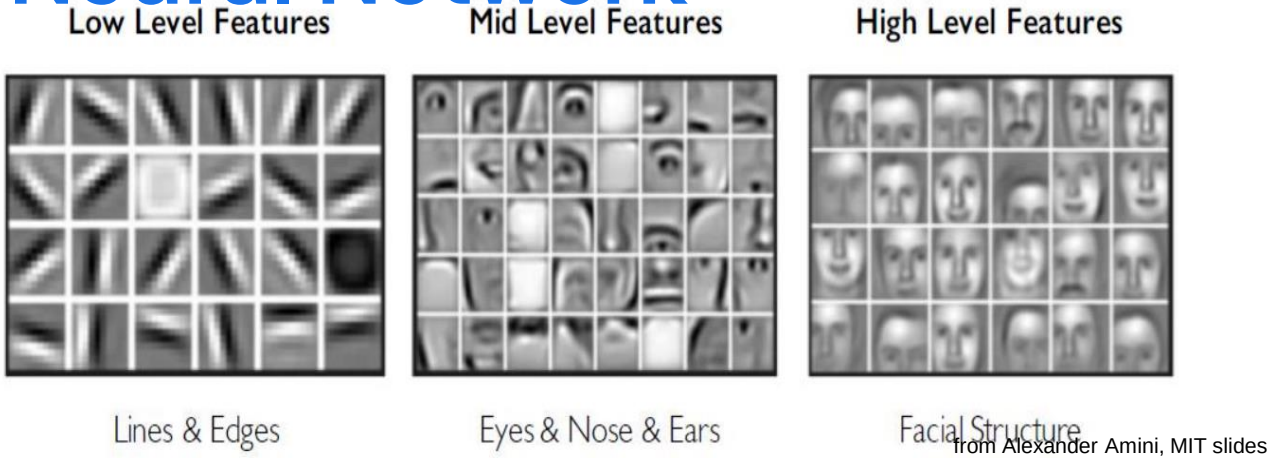
Deep Neural Network



By adding more hidden layers, the network can learn higher-level features and recognize complex patterns better than a shallow network.

Deeper networks power modern AI applications like self-driving cars, chatbots, and medical diagnosis because they learn better from data.

Deep Neural Network



from Alexander Amini, MIT slides

Example: Recognizing a Face in a Photo

- **First Layer (Edges & Lines 🏗️)** → Detects very basic features like edges or curves.
- **Second Layer (Shapes 📐)** → Combines edges to recognize shapes like circles, triangles, or rectangles.
- **Third Layer (Objects 👀)** → Builds on shapes to identify specific features like eyes, nose, or lips.
- **Final Layer (Complete Recognition 😊)** → Understands that all these features together make a face!

Each layer learns something **new and builds on the previous laye**r, which makes deep networks powerful for recognizing patterns, objects, and even language!

There is a great variety of Neural Network architectures out there

- **FeedForward NN** → The simplest type of artificial neural network where the connections between the nodes do not form a cycle.
- **Convolutional NN** → Primarily used for processing data that has a grid-like topology, such as images.
- **Recurrent NN** → Designed for sequence prediction problems and time series data, capable of retaining information from previous inputs by using loops within the network.
- **Long-Short Term Memory Network** → A type of RNN that is capable of learning long-term dependencies, commonly used in sequence prediction.

Sequence Models – Why We Need Them

What is Sequence Data?

- Information that comes in a specific order over time
- **Examples:**
  - Sentences in a paragraph
  - Words in speech
  - Daily stock prices

Dependence on Previous Data

- Many tasks require knowing past information to make accurate predictions
- **Examples:**
  - Predicting the next word in a sentence requires knowledge of prior words
  - Forecasting stock prices depends on past price trends

Sequence Models – Why We Need Them

What is Sequence Data?

If you know the past …

… you may be able to predict the future.

Sequence Models – Why We Need Them

Why Sequence Models are Better

- Traditional models don’t capture time-based dependencies well
- **Sequence Models:**
  - **RNNs (Recurrent Neural Networks)**
  - **LSTMs (Long Short-Term Memory networks)**
- **Advantages:**
  - Can "remember" past information
  - Better at handling sequential data for tasks like language modeling and time-series forecasting

Recurrent Neural Network (RNN)

Recurrent Neural Network

RNNs are good at modeling sequence data.

Video analysis

Stock price prediction

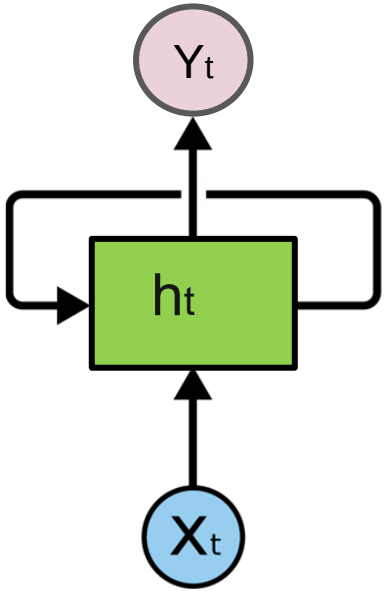
Speech recognition

Machine translation

ECG diagnosis

Recurrent Neural Network (contd.)

It is a special type of neural network designed to handle **sequential data**, meaning data where the order of inputs matters—like sentences in language, stock prices over time, or even music notes in a melody.



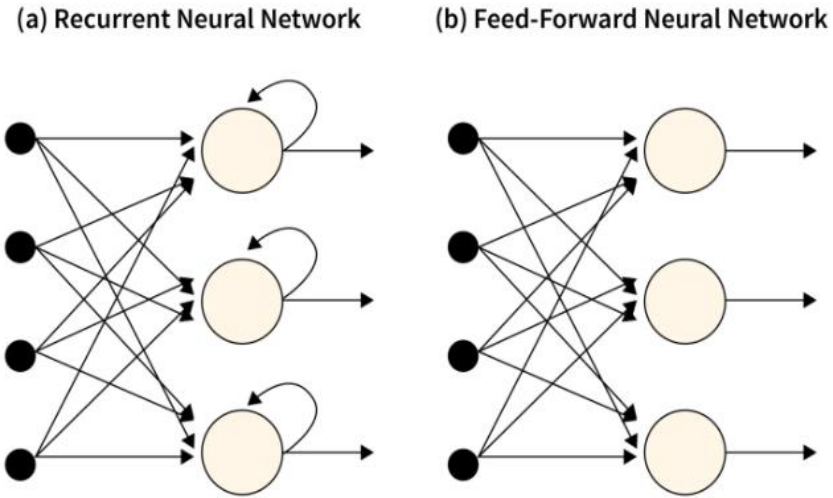
An RNN has a **loop** that allows it to **pass information from previous steps forward.**

This loop is called a **hidden state** (denoted as $\mathbf{h_t}$).

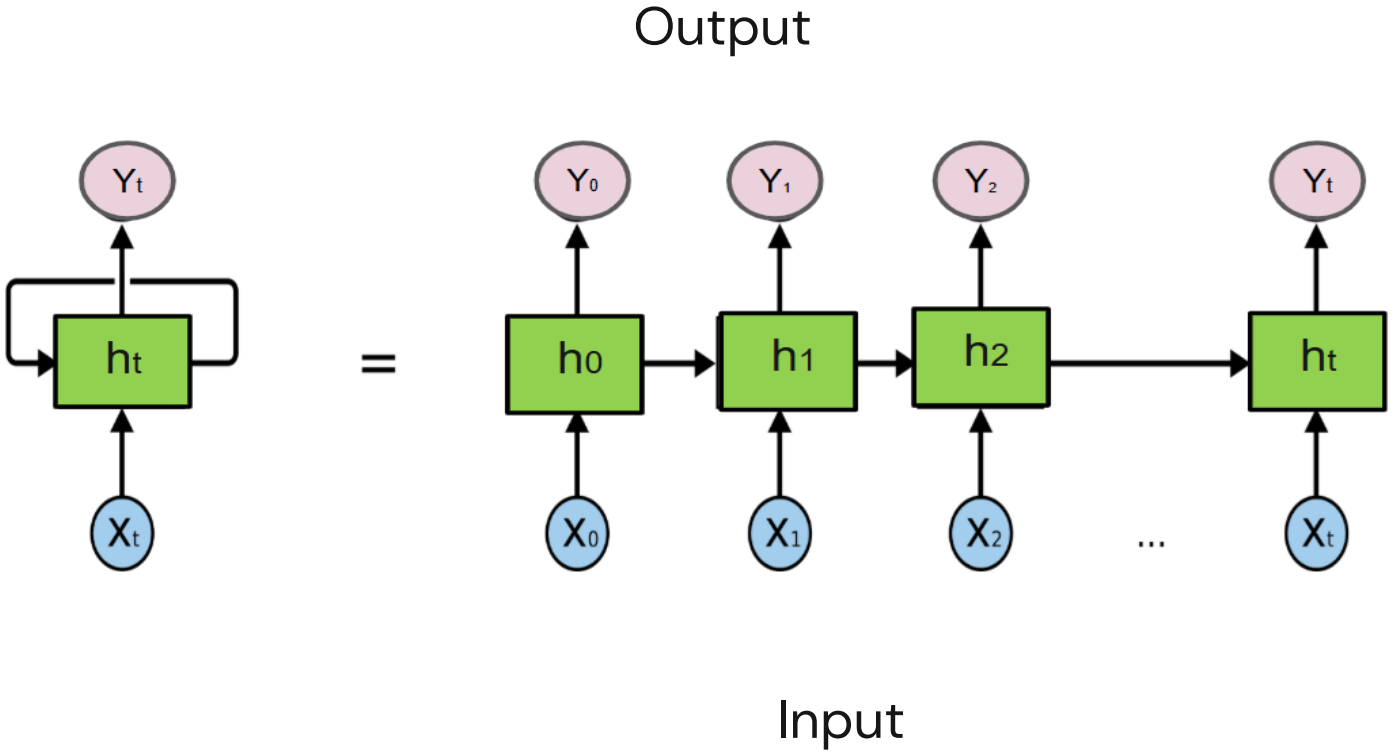
Recurrent Neural Network (contd.)

🔷 **How is RNN Different from a Normal Neural Network?**

Unlike traditional neural networks (which process inputs independently), RNNs **remember past inputs using loops** within their structure. This memory helps them **understand patterns over time**.

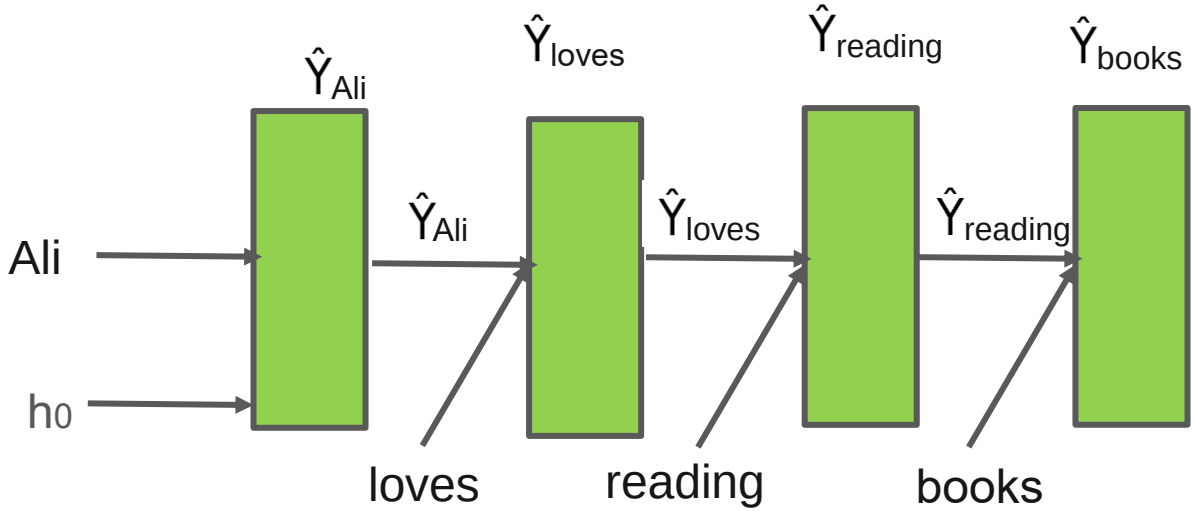


Unrolled RNN



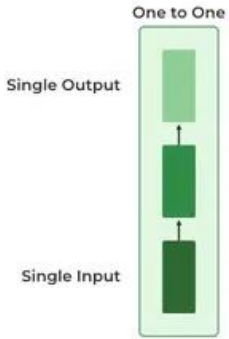
Example RNN

Ali loves reading books

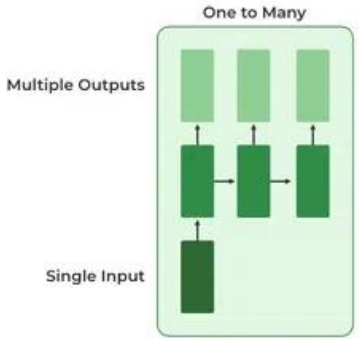


Types of RNN

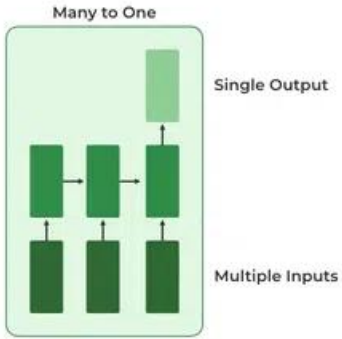
1. One-to-One RNN



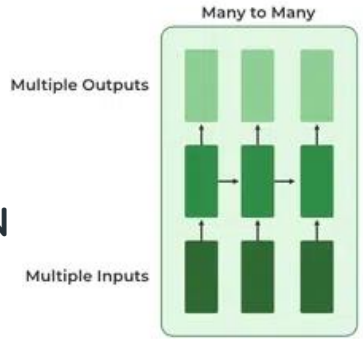
2. One-to-Many RNN



3. Many-to-One RNN



4. Many-to-Many RNN

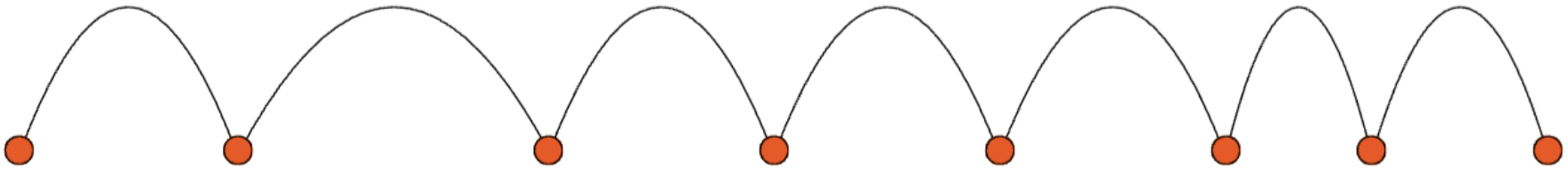


Source: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

How did they predict text?

RNNs predict output by assessing the relationship of a word with its immediate neighbors

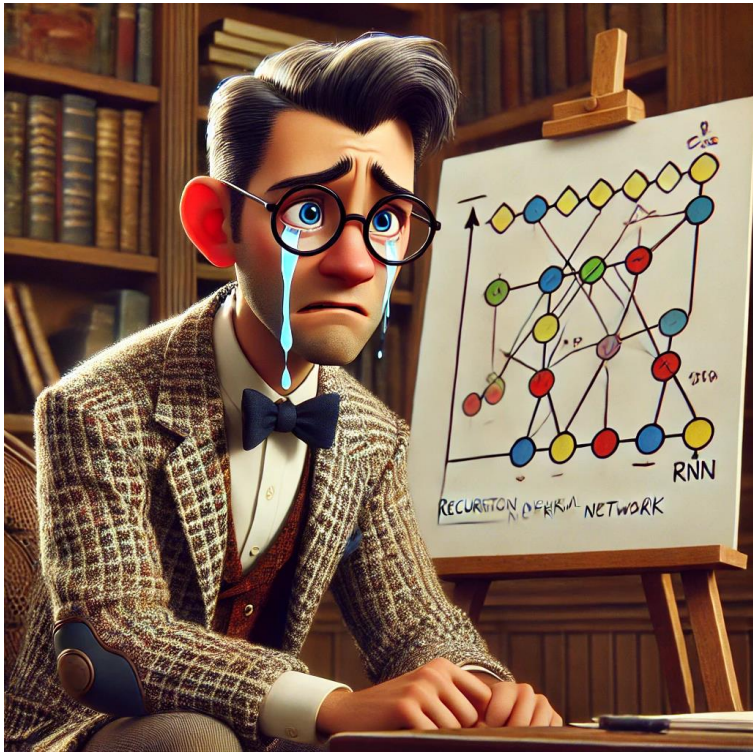
Source: Data Science Dojo!



The teacher taught the student with the book.

BUT...

Recurrent Neural Nets had two major problems...



1. Difficulty in handling long sequences

RNNs struggle to capture information from distant points in sequences, which makes it difficult to learn long-range dependencies in data such as text.

This problem is called Vanishing gradient problem.

Source: Data Science Dojo!

2. Computational Complexity

Training RNNs can be quite resource- intensive, particularly with extended sequences. This is attributed to the sequential processing of each input by the network, a method that can be time-consuming.

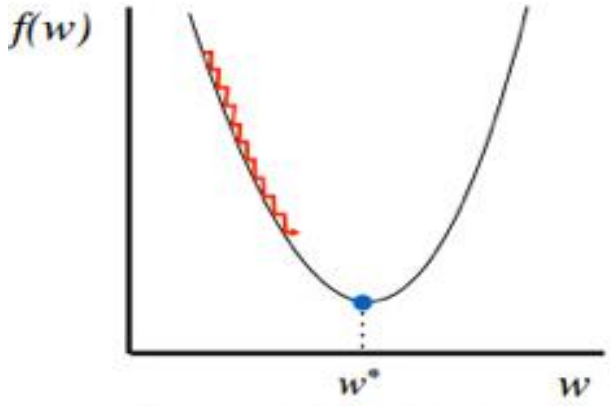
Source: Data Science Dojo!

Vanishing and Exploding Gradients

When processing long sequences, gradients can either become too small (vanishing) or too large (exploding), making it difficult for the model to learn effectively.

Vanishing Gradient

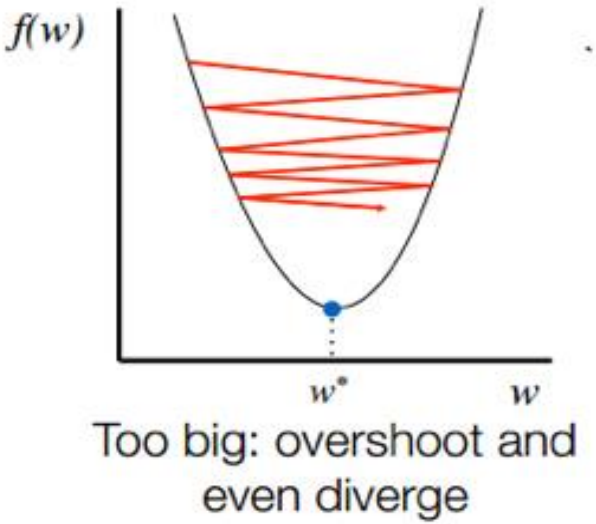
As the backpropagation algorithm advances downwards(or backward) from the output layer towards the input layer, the **gradients often get smaller and smaller and approach zero**, eventually leaving the **weights of the initial or lower layers nearly unchanged**. As a result, the gradient descent never converges to the optimum. This is known as the **vanishing gradients problem.**



Too small: converge very slowly

Exploding Gradient

When the **gradients keep getting larger in some cases as the backpropagation algorithm progresses.** This, in turn, causes **large weight updates** and causes the gradient descent to diverge. This is known as the **exploding gradient problem.**



https://srdas.github.io/DLBook/GradientDescentTechniques.html

Vanishing & Exploding Gradients

- $h_{i+1} = A(h_i, x_i)$
- $A(h_i, x_i) = \mathbb{W}x_i + Zh_i$
- $h_3 = A(A(A(h_0, x_0), x_1), x_2)$
- $h_N = A(A(\cdots A(h_0, x_0), x_1), x_2 \cdots), x_{N-2}), x_{N-1})$
- $h_N = \mathbb{W}x_{N-1} + Z\mathbb{W}x_{N-2} + Z^2\mathbb{W}x_{N-3} + \cdots + Z^{N-1}\mathbb{W}x_0 + Z^N h_i$

Multiplying matrix N times!

Long Short Term Memory (LSTM)

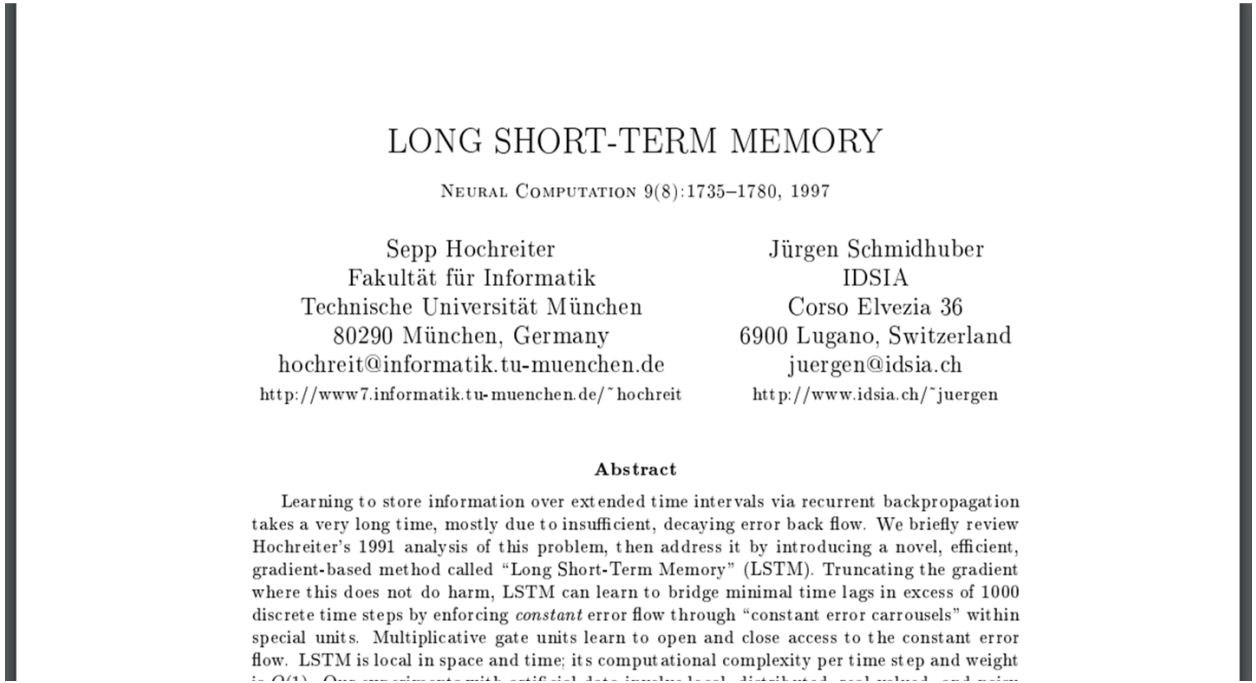
RNN's Problem of Difficulty in handling long sequences

RNNs struggle to capture information from distant points in sequences, which makes it difficult to learn long-range dependencies in data such as text.

Solved by LSTM

Source: Data Science Dojo!

LSTM to the Rescue



LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

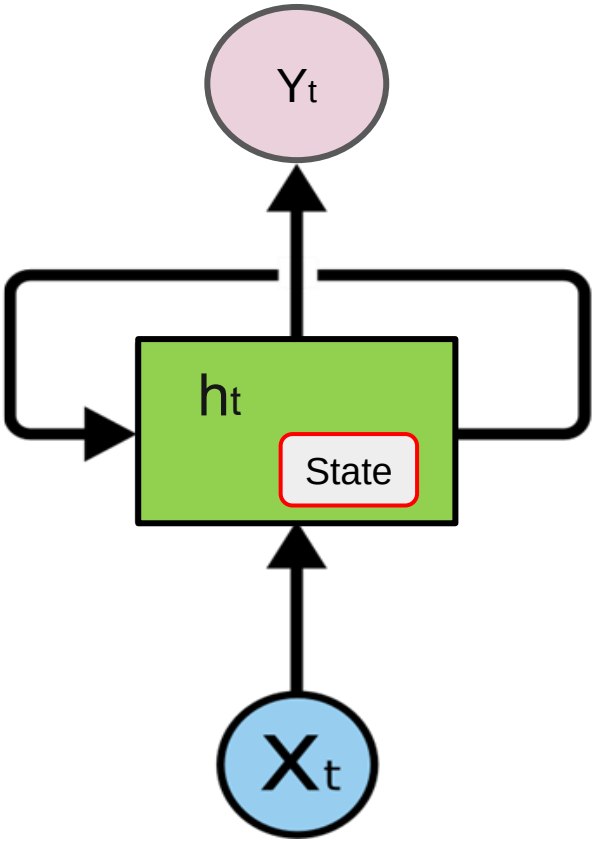
Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Abstract

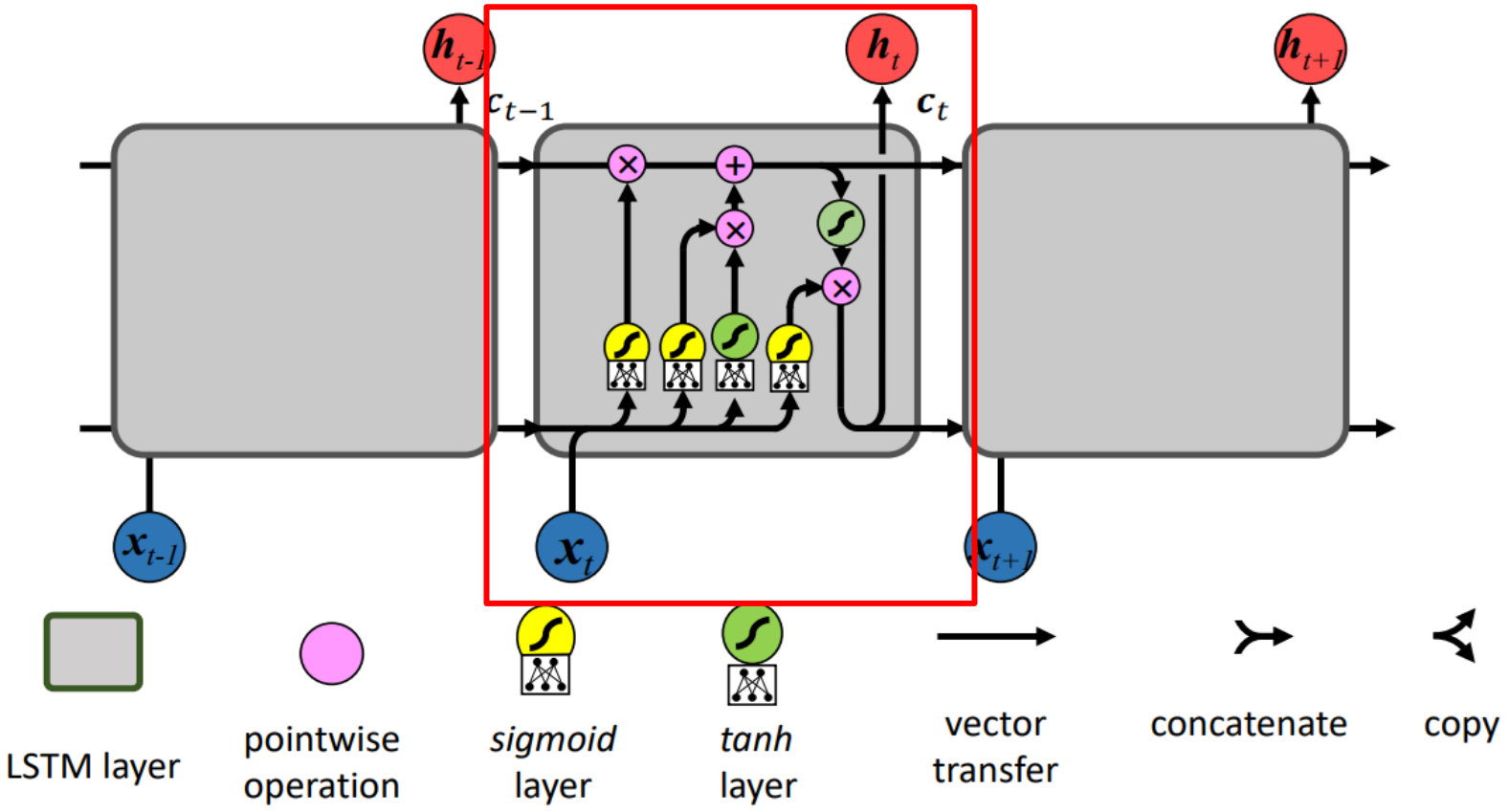
Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight

DOI: https://doi.org/10.1162/neco.1997.9.8.1735

Long Short Term Memory (LSTM)



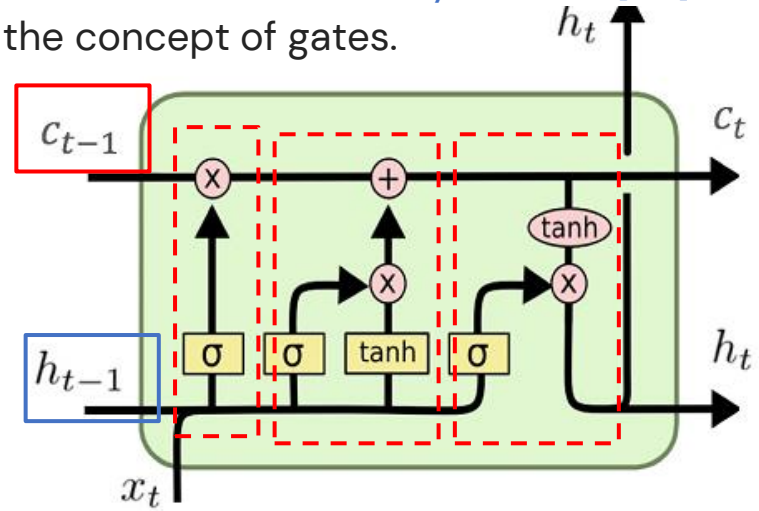
LSTM Interacting Layers



LSTM: The Core Idea

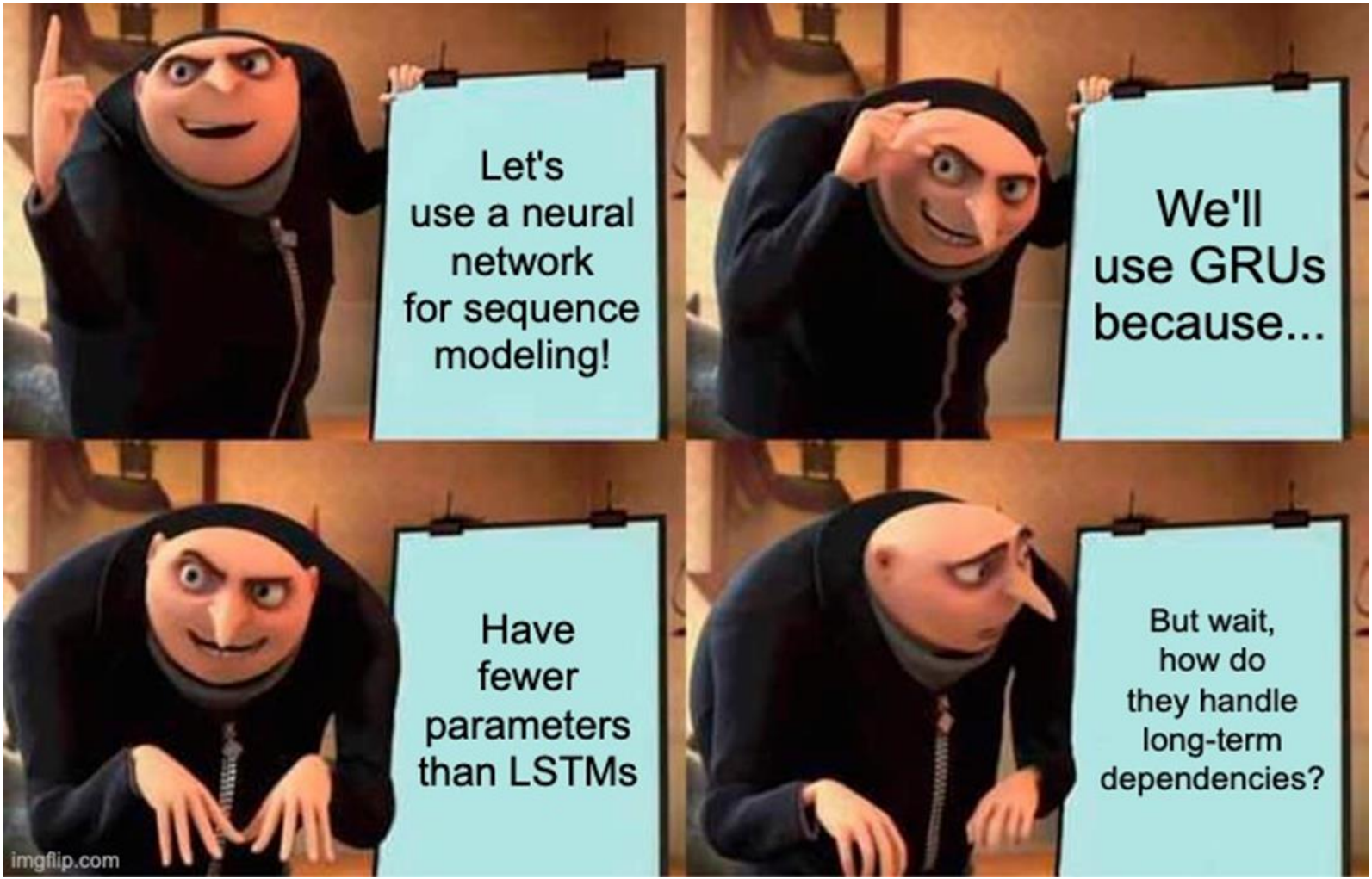
LSTMs deal with both Long Term Memory (LTM / c[t-1]) and Short Term Memory (STM / h[t-1]) and for making the calculations simple and effective it uses the concept of gates.

- **Forget Gate:** LTM goes to forget gate, and it forgets information that is not useful.
- **Input Gate:** Event (sequence input / x[t]) and STM are combined so that necessary information that we have recently learned from STM can be applied to the current input.
- **Remember Gate:** LTM information that we haven't forgotten and STM and Event are combined in Remember gate, which works as updated LTM.
- **Output Gate:** This gate also uses LTM, STM, and Event to predict the output of the current event, which works as an updated STM



LSTM
(Long-Short Term Memory)

Gated Recurrent Units (GRU)

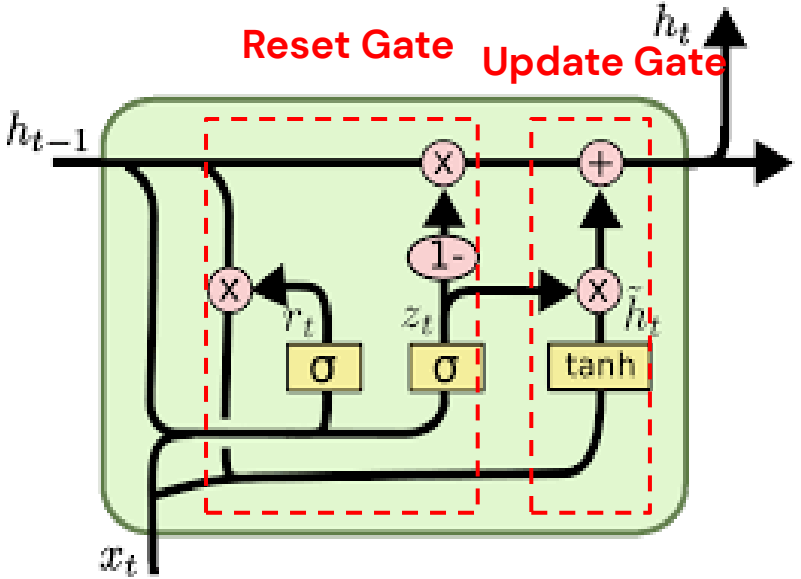

Let's use a neural network for sequence modeling!
We'll use GRUs because...
Have fewer parameters than LSTMs
But wait, how do they handle long-term dependencies?
imgflip.com

Gated Recurrent Units (GRU)

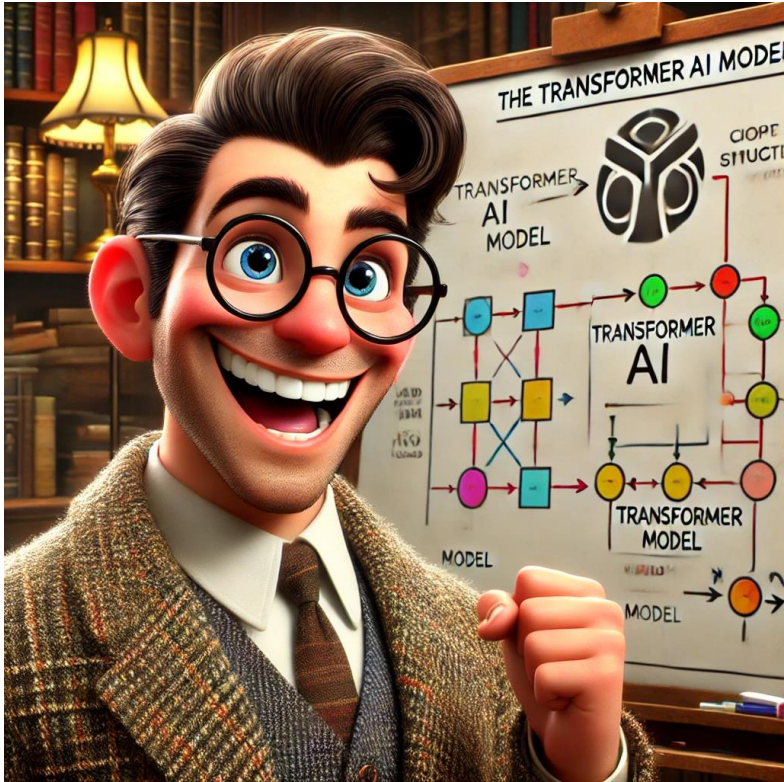
GRUs are similar to the LSTM networks. GRU is a kind of newer version of RNN.

However, there are some differences between GRU and LSTM.

- GRU doesn't contain a cell state
- GRU uses its hidden states to transport information
- **It Contains only 2 gates(Reset and Update Gate)**
- GRU is faster than LSTM
- GRU has lesser tensor's operation that makes it faster



Transformers managed to solve these issues with more accuracy and less cost



RNN's Problem of Computational Complexity

Training RNNs can be quite resource- intensive, particularly with extended sequences. This is attributed to the sequential processing of each input by the network, a method that can be time-consuming.

Transformers (like GPT, BERT)

Source: Data Science Dojo!

Thank You