Topic 7a

Recurrent Neural Network

(Basics)

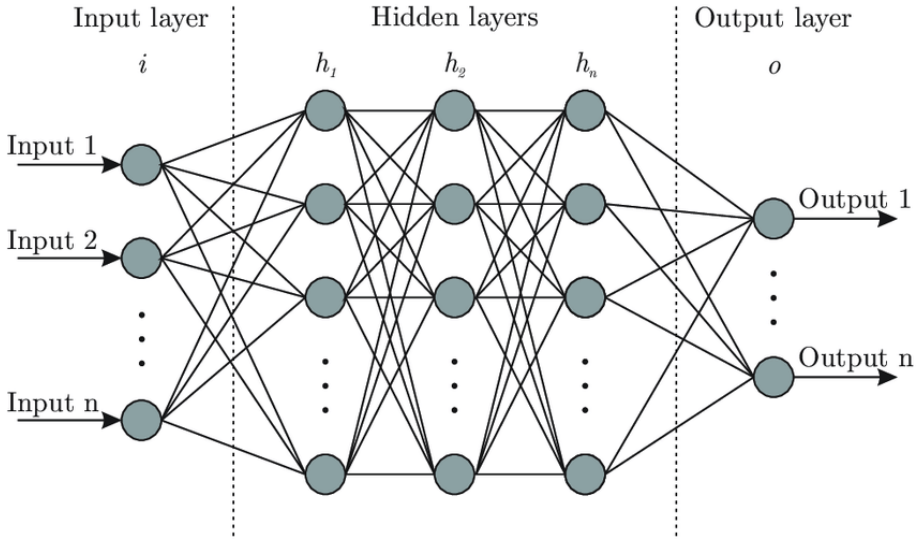
CSE465: Pattern Recognition and Neural Network
Sec: 3
Faculty: Silvia Ahmed (SvA)
Summer 2025

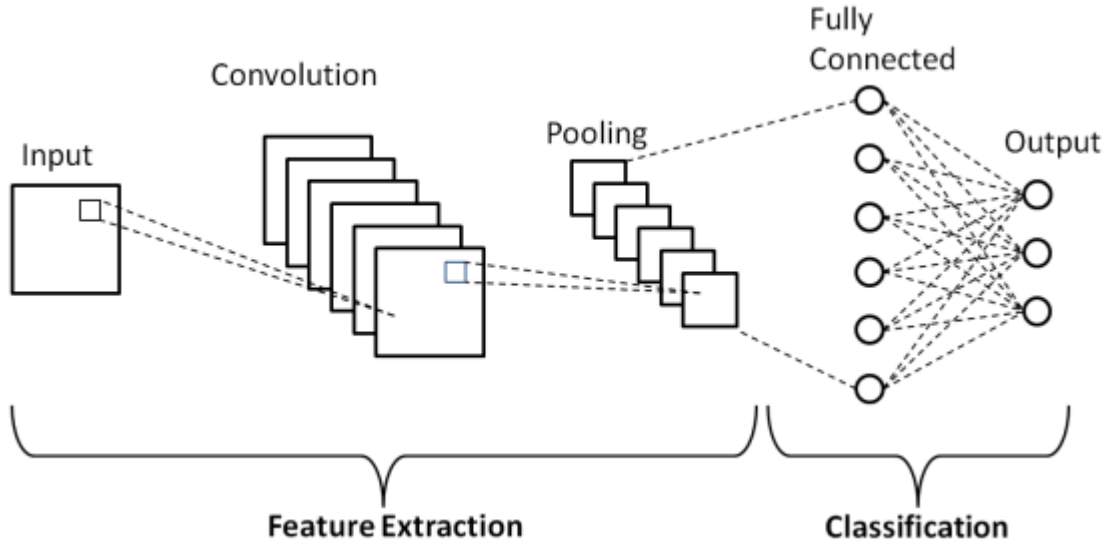
Topics

1. What is Sequential Data?
    1. Handling sequential data with ANN and CNN
    2. Limitations of ANN and CNN for sequential data
2. RNN
    1. Applications of RNN
    2. Architecture of RNN
    3. Forward Propagation
    4. Types of RNN
    5. Backward Propagation Through Time (BPTT)

ANN and CNN so far..

A	HR	E	S	C	BP	T
20	60	0	1	100	100	1
50	100	0	0	200	125	1
60	70	1	1	195	110	0
33	80	1	1	203	115	0
45	75	0	1	179	119	1
18	65	1	0	150	122	1
91	88	1	1	188	135	0
77	101	1	1	220	100	1
23	63	0	1	215	125	1





Sequential Data

- Sequential data is a type of data where the order of elements matters.
- Each piece of information depends on previous and/or future elements in the sequence.
- Unlike regular data, where items can be shuffled without losing meaning, sequential data follows a specific order—changing this order alters its meaning.

Example

- Text (Natural Language Processing)
  - Example: *"I love programming"* vs. *"Programming love I“*
  - The meaning depends on the word order.
- Time Series Data
  - Example: Daily stock prices, weather temperatures, or heart rate over time.
  - Yesterday’s stock price affects today's price.
- Speech and Audio
  - Example: A sentence in speech must be heard in order; otherwise, it becomes gibberish.
- Videos
  - Frames in a video must play in the correct order for the story to make sense.

Sequential vs Other data types

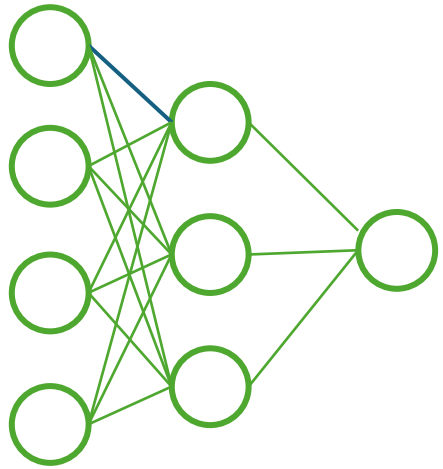
Feature	Sequential Data	Non-Sequential Data
Order Matters?	☑ Yes	✗ No
Examples	Text, speech, stock prices	Images, customer records, tabular data
Shuffling Allowed?	✗ No (Changes meaning)	☑ Yes (No impact on meaning)
Memory Required?	☑ Yes (Past elements influence the present)	✗ No (Each element is independent)

Is ANN or CNN suitable for Sequential data?

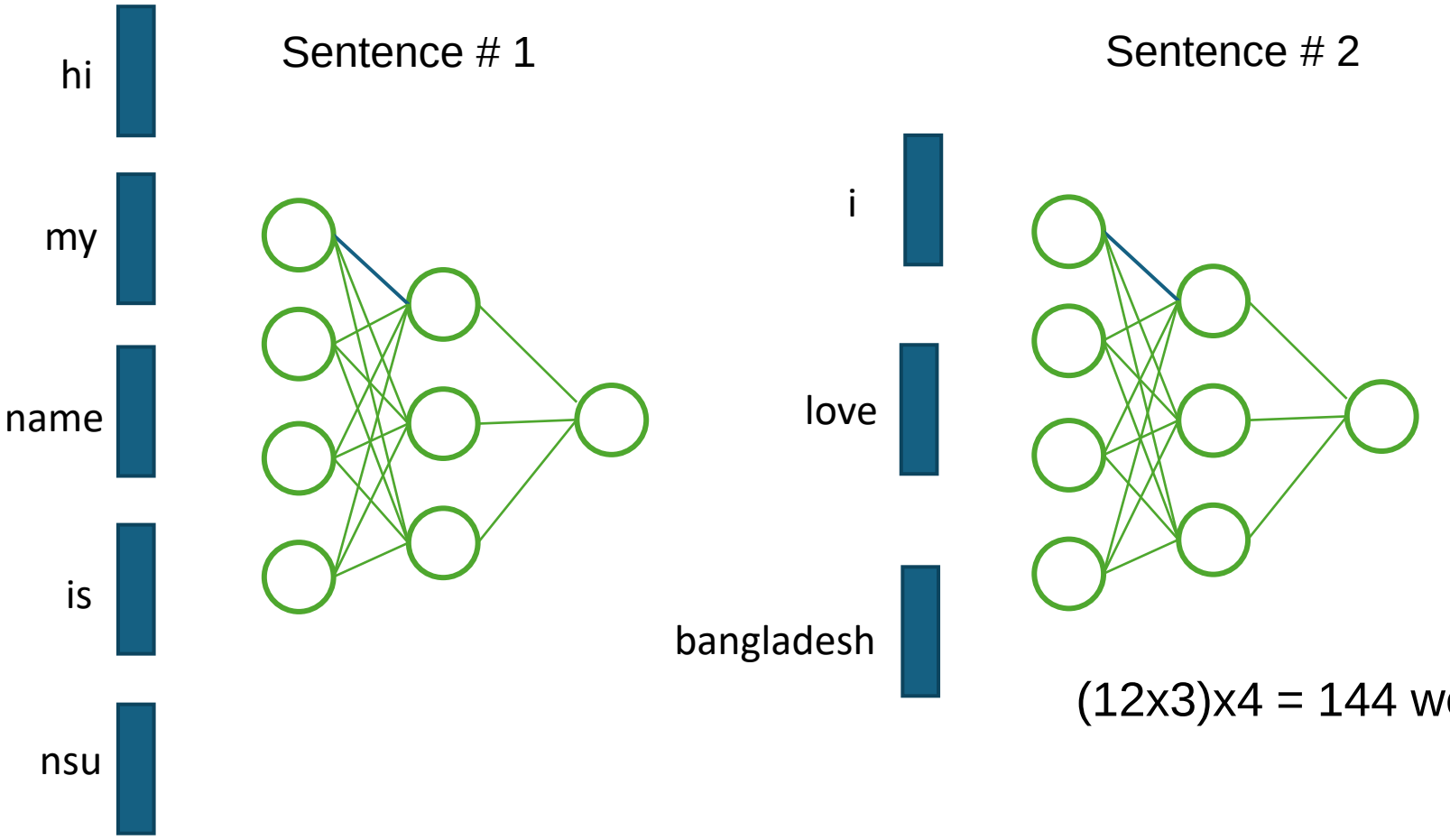
Input	Output
Hi my name is NSU	0
I love Bangladesh	0
Bangladesh won the match	1

- Vectorize (OHE, Bag Of Words, TFIDF, Word2Vec)
- One Hot Encoding:
  - Unique words: 12
  - Take a vector of 12 size
  - `hi: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
  - `my: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

ANN



(12x5)x4 = 240 weights

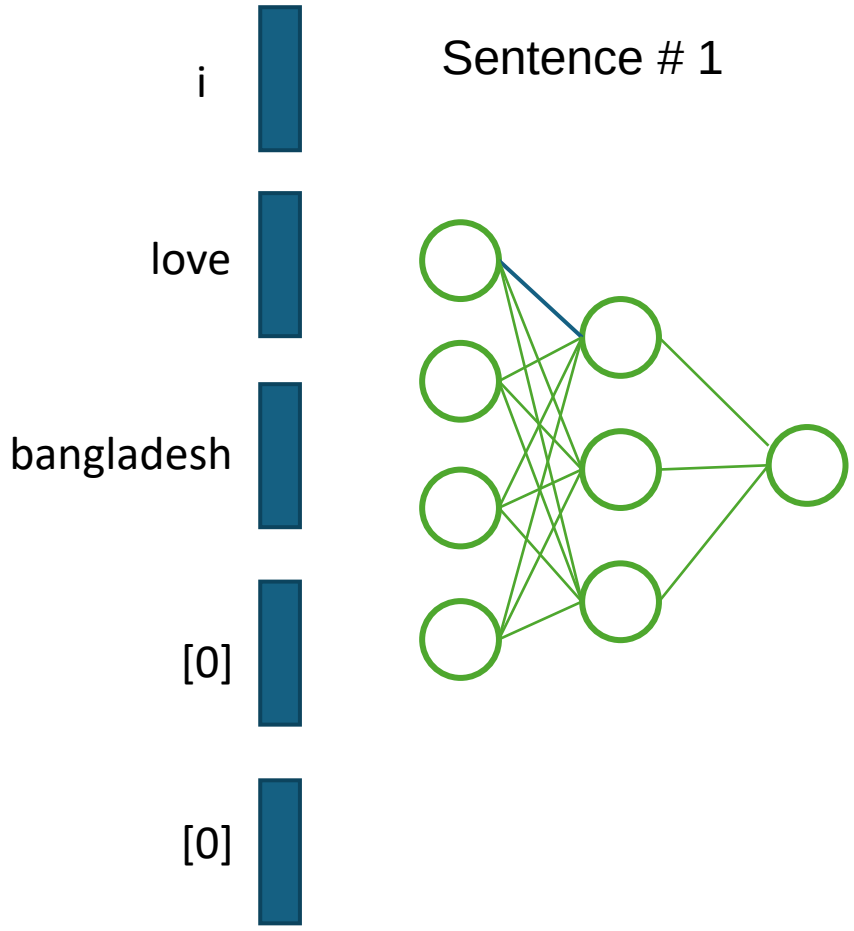

hi
my
name
is
nsu
Sentence # 1
Sentence # 2
i
love
bangladesh
Problem:
Varying input size
(12x3)x4 = 144 weights

Problem:
Varying input size

Solution:
Zero Padding

Dictionary: 10000 words
Sentence: 100 words
Input weights : 1,000,000

New problem:
Unnecessary computation



Overall problems using ANN and CNN

- Text input → Varying size
- Zero padding → Unnecessary computation
- Prediction Problem
- Totally disregarding the sequence information

- Recurrent Neural Network:
  - ☑ **Best suited** for sequential data because:
    - RNNs have loops that allow them to "remember" past information, making them ideal for tasks where order matters.
    - Example: In text generation, RNNs can remember previous words to generate coherent sentences.

Application of RNN

- Natural Language Processing (NLP)
  - **Machine Translation** (e.g., Google Translate)
    - Converts text from one language to another while maintaining context.
  - **Text Generation** (e.g., Chatbots, AI Writing Assistants)
    - Predicts the next word in a sentence to generate human-like text.
  - **Sentiment Analysis** (e.g., Analyzing Customer Reviews)
    - Determines whether a review is positive, negative, or neutral.
  - **Speech-to-Text** (e.g., Voice Assistants like Siri, Alexa)
    - Converts spoken words into written text.

Application of RNN (contd.)

- Time Series Forecasting
  - Stock Price Prediction
    - Predicts future stock prices based on past trends.
  - Weather Forecasting
    - Uses past weather patterns to predict future weather conditions.
  - Sales Forecasting
    - Helps businesses predict future sales based on historical data.

Application of RNN (contd.)

- Speech and Audio Processing
  - **Speech Recognition** (e.g., Google Voice Search)
    - Converts spoken language into text.
  - **Music Generation** (e.g., AI-Generated Music)
    - AI composes music by learning patterns in existing compositions.
  - Voice Cloning
    - Generates human-like speech from a small sample of a person’s voice.

Application of RNN (contd.)

- Video and Image Processing
  - Lip Reading AI
    - Uses RNNs to predict words by analyzing lip movements in videos.
  - **Handwriting Recognition** (e.g., Digitizing Handwritten Notes)
    - Converts handwritten text into digital text.
  - **Action Recognition in Videos** (e.g., Surveillance Systems)
    - Identifies human activities like running, walking, or fighting in security footage.

Application of RNN (contd.)

- Healthcare Applications
  - Disease Progression Prediction
    - Predicts how a disease (e.g., Parkinson’s, Alzheimer’s) will develop over time.
  - Medical Report Generation
    - Automatically summarizes medical findings from patient data.
  - ECG and EEG Signal Analysis
    - Helps detect heart or brain abnormalities by analyzing sequential health data.

Application of RNN (contd.)

- Autonomous Systems
  - **Self-Driving Cars** (e.g., Tesla Autopilot)
    - RNNs help predict pedestrian movements and vehicle behavior.
  - Robotics
    - Enables robots to understand and react to sequential actions.
- Chatbots & Virtual Assistants
  - **Conversational AI** (e.g., ChatGPT, customer service bots)
    - Helps machines understand and respond in human-like conversations.

Application of RNN (contd.)

- Fraud Detection & Anomaly Detection
  - Credit Card Fraud Detection
    - Identifies unusual spending patterns in transactions.
  - Cybersecurity Threat Detection
    - Detects suspicious network activity to prevent cyberattacks.
- Gaming and AI Agents
  - **AI Players in Games** (e.g., OpenAI's Dota 2 Bot)
    - AI uses RNNs to predict opponent moves and react strategically.
- DNA Sequence Analysis
  - Genomics & Drug Discovery
    - Helps in analyzing genetic sequences for medical research.

Topic 4: Recurrent Neural Network

RNN Architecture

Data for RNN (Keras)

- (timesteps, input_features)
- Vocabulary:
  - No of unique words: 5
  - movie: [1 0 0 0 0]
  - was: [0 1 0 0 0]
  - good: [0 0 1 0 0]
  - bad: [0 0 0 1 0]
  - not: [0 0 0 0 1]
  - Review 1: [1 0 0 0 0], [0 1 0 0 0], [0 0 1 0 0] → (3, 5)
  - Review 2: [1 0 0 0 0], [0 1 0 0 0], [0 0 0 1 0] → (3, 5)
  - Review 3: [1 0 0 0 0], [0 1 0 0 0], [0 0 0 0 1], [0 0 1 0 0] → (4, 5)

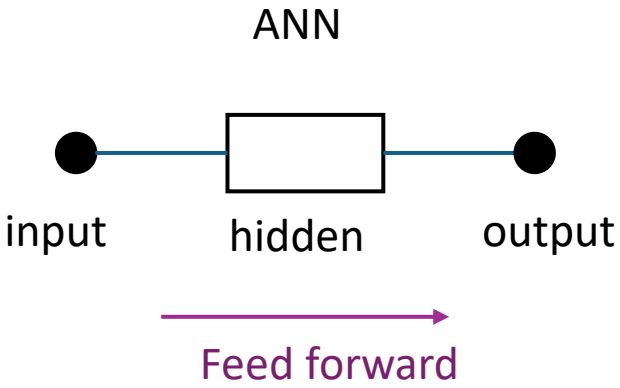
Review	Sentiment
Movie was good	1
Movie was bad	0
Movie was not good	0

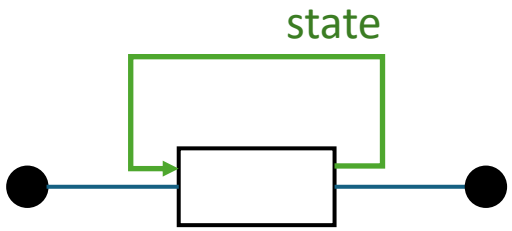
How RNN works?

	Review	Sentiment
X_1	Movie was good	1
	X_{11} X_{12} X_{13}	
X_2	Movie was bad	0
X_3	Movie was not good	0

Vocabulary:

- movie: [1 0 0 0 0]
- was: [0 1 0 0 0]
- good: [0 0 1 0 0]
- bad: [0 0 0 1 0]
- not: [0 0 0 0 1]





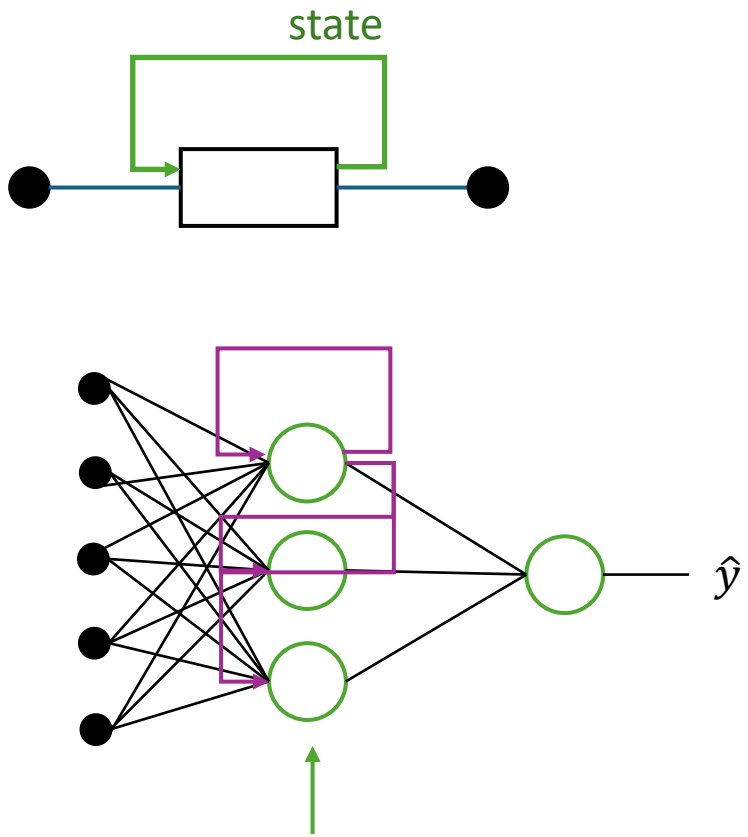
t = 1: X_{11}
t = 2: X_{12}
t = 3: X_{13}

How RNN works? (contd.)

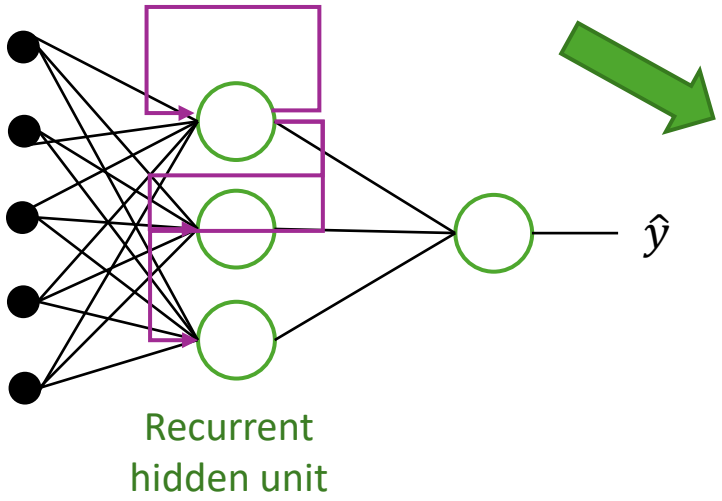
	Review	Sentiment
X_1	Movie was good	1
	X_{11} X_{12} X_{13}	
X_2	Movie was bad	0
X_3	Movie was not good	0

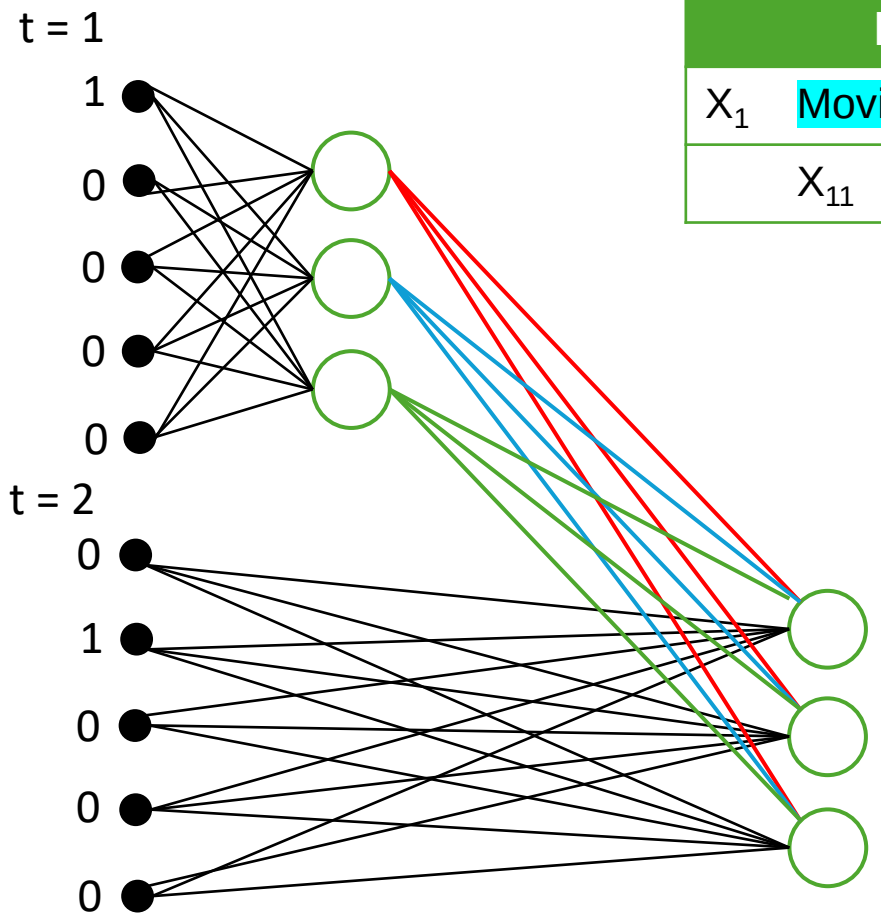
Vocabulary:

- movie: [1 0 0 0 0]
- was: [0 1 0 0 0]
- good: [0 0 1 0 0]
- bad: [0 0 0 1 0]
- not: [0 0 0 0 1]



How RNN works? (contd.)





	Review			Sentiment
X_1	Movie	was	good	1
	X_{11}	X_{12}	X_{13}	

Number of weights and biases:

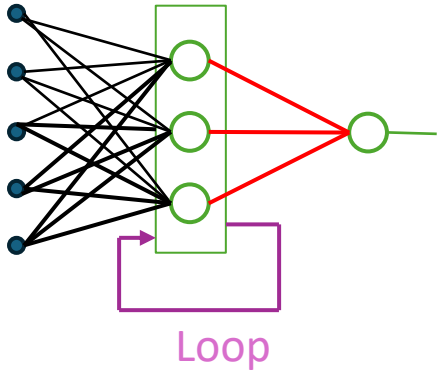
- Input → hidden: 5 x 3 = 15
- Hidden feedback: 3 x 3 = 9
- Hidden → output: 3 x 1 = 3
- Biases: 3 + 1 = 4

For t = 1, the feedback will be either zeros or random numbers

RNN Forward Propagation

Review	Sentiment
X_{11}, X_{12}, X_{13}	1
X_{21}, X_{22}, X_{23}	0
$X_{31}, X_{32}, X_{33}, X_{34}$	0

X_{ij}: Vectors of dim 5



W_i : (5,3)

Forward → unfolding/unrolling through time

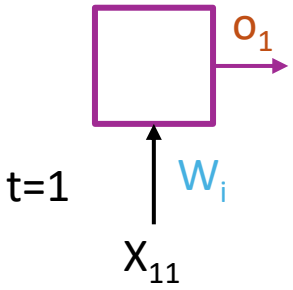
Activation: tanh/ReLU

At t = 1:

$X_{11}W_i + b$ → $f(X_{11}W_i + b)$

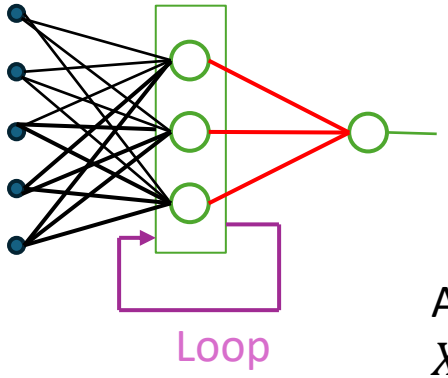
(1,5),(5,3) → O_1

(1,3) (1,3) → (1,3)



RNN Forward Propagation

Review	Sentiment
X_{11}, X_{12}, X_{13}	1
X_{21}, X_{22}, X_{23}	0
$X_{31}, X_{32}, X_{33}, X_{34}$	0



W_i : (5,3)

Forward → unfolding/unrolling through time

Activation:
tanh/ReLU

At t = 2:

$X_{12}W_i + b + o_1W_h$

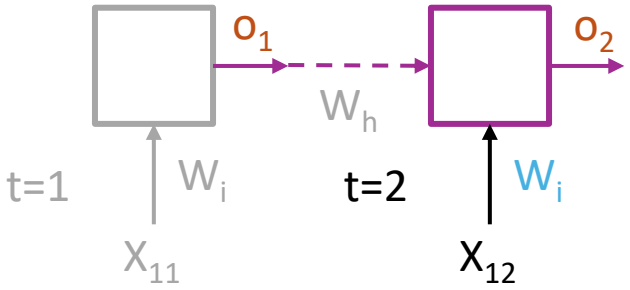
(1,3),(3,3)

(1,3)

$f(X_{12}W_i + o_1W_h + b)$

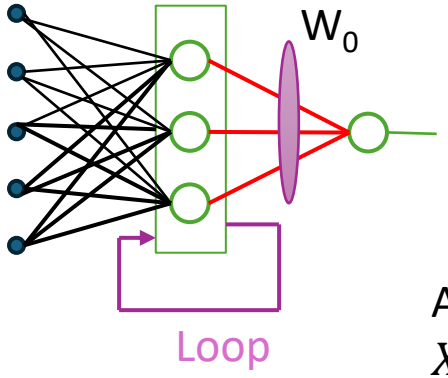
O_2

(1,3)



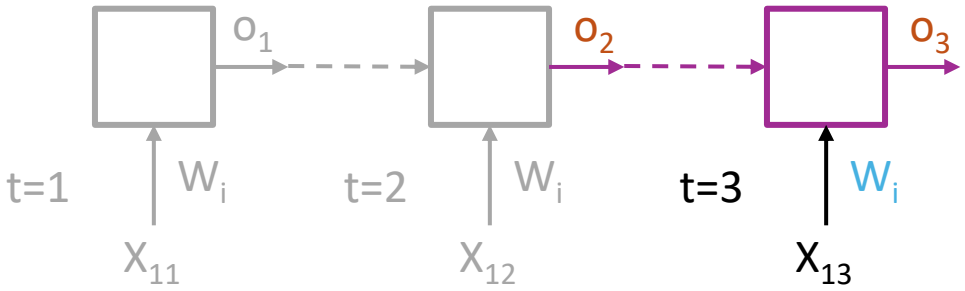
RNN Forward Propagation

Review	Sentiment
X_{11}, X_{12}, X_{13}	1
X_{21}, X_{22}, X_{23}	0
$X_{31}, X_{32}, X_{33}, X_{34}$	0



W_i : (5,3)

Forward → unfolding/unrolling through time



At t = 3:

$X_{13}W_i + b + o_2W_h$

(1,3),(3,3)

(1,3)

Activation: tanh/ReLU

$f(X_{13}W_i + o_2W_h + b)$

O_3

(1,3)

After t = 3:

$o_3W_2 + b$

(1,3),(3,1)

(1,1)

Activation: sigmoid

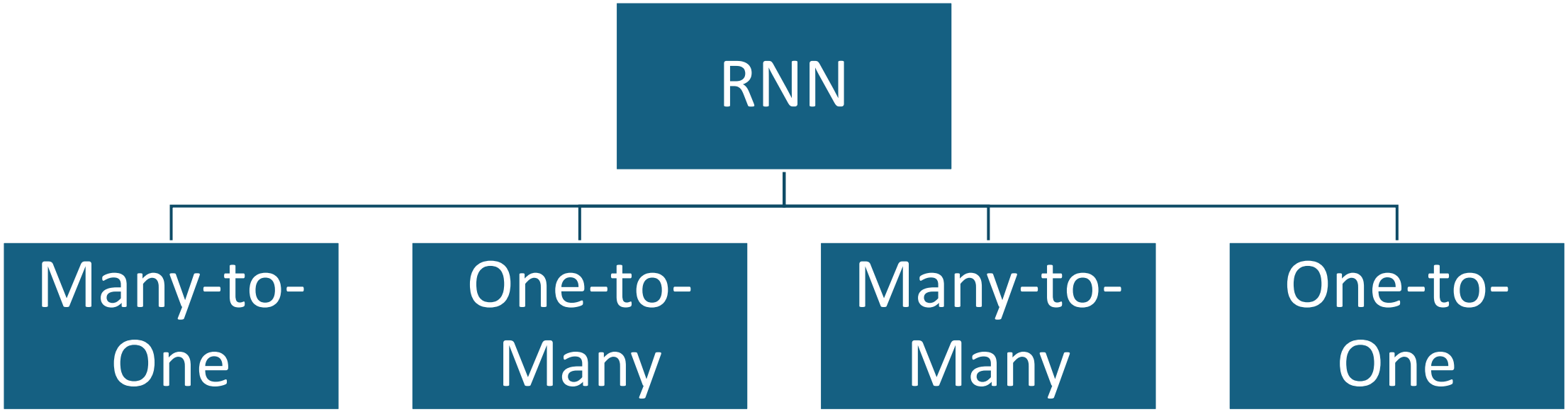
$f(o_3W_0 + b)$

(1,1)

Topic 4: Recurrent Neural Network

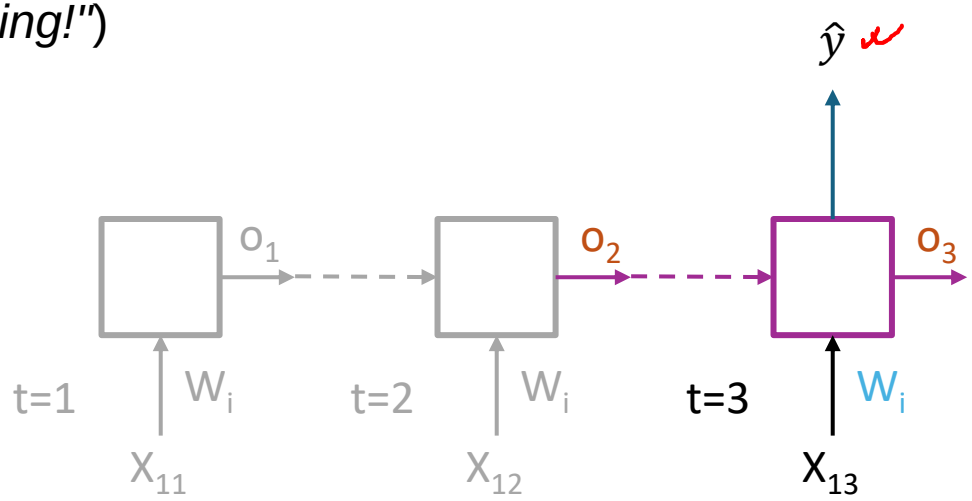
Types of RNN Architecture

Types of RNN



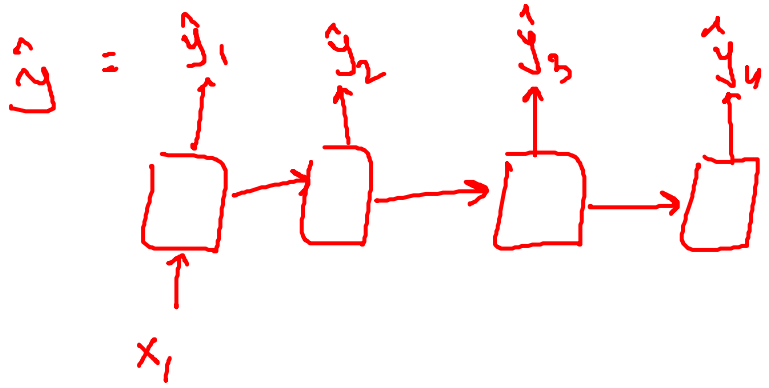
Many-to-One (M→1) RNN

- Multiple input time steps, but only one output.
- The RNN processes the entire sequence and generates a single output based on all past inputs.
- Example Use Cases:
  - Sentiment Analysis:
    - Input: A sentence (*"I love this movie, it was amazing!"*)
    - Output: A single sentiment label (e.g., **positive**)
  - Spam Detection:
    - Input: An email with multiple words
    - Output: A single label (**spam or not spam**)
  - Stock Price Prediction
    - Input: Past **30 days** of stock prices
    - Output: Stock price on **day 31**



One-to-Many (1→M) RNN

- A single input produces a sequence of outputs over time.
- Used for **sequence generation** tasks where output unfolds over time.
- Example Use Cases
  - Music Generation:
    - Input: A starting note
    - Output: A full melody
  - Image Captioning
    - Input: A single image
    - Output: A sequence of words describing the image
  - Text-to-Speech (TTS)
    - Input: A single word
    - Output: Multiple sound waves forming a spoken version



Many-to-Many (M→M) RNN

- This category has **two variations**:
  - **Equal Length Many-to-Many** (M→M)
  - **Unequal Length Many-to-Many** (M→M with different input/output lengths)
- (a) Many-to-Many (Equal Input & Output Lengths):
  - Every input time step corresponds to an output at the same time step.
  - Example:
    - Parts of Speech Tagging
      - Input: *"My name is Silvia"*
      - Output: *(pronoun, noun, verb, noun)*
    - Named Entity Recognition (NER)
      - Input: *"Alice lives in Paris."*
      - Output: *(Name, Verb, Location)*

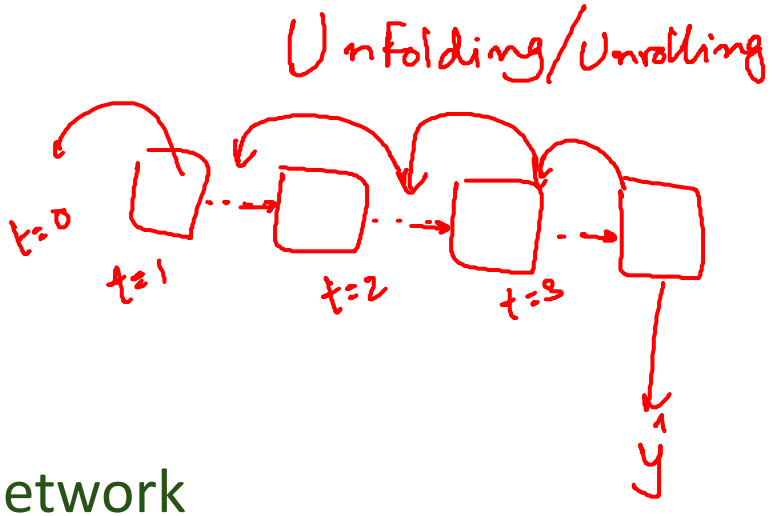
Preposition

NLP

Unequal length:
Machine Translation

One-to-One (1→1) RNN

- Each input corresponds to a single output.
- This is the simplest form of an RNN and behaves similarly to a **traditional feedforward neural network (FNN)**.
- Example Use Cases:
  - Simple Binary Classification
    - Input: A single image
    - Output: Label (e.g., "Cat" or "Dog")
- Why Not Use RNN for One-to-One?
  - Since RNNs are designed for **sequential data**, using them for **static** one-to-one tasks is inefficient.
  - A simple **Artificial Neural Network (ANN)** or **Convolutional Neural Network (CNN)** (for images) is better suited.



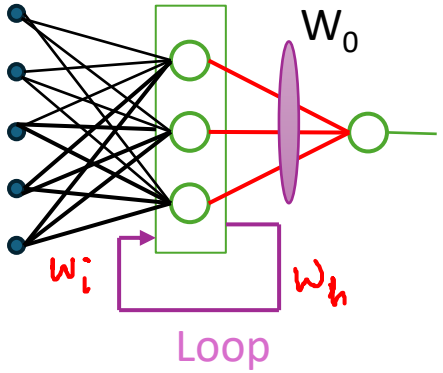
Topic 4: Recurrent Neural Network

Backpropagation Through Time (BPTT)

Example: M→1 Sentiment Analysis

Review	Sentiment
X_{11}, X_{12}, X_{13}	1
X_{21}, X_{22}, X_{23}	0
$X_{31}, X_{32}, X_{33}, X_{34}$	0

X_{ij}: Vectors of dim 5

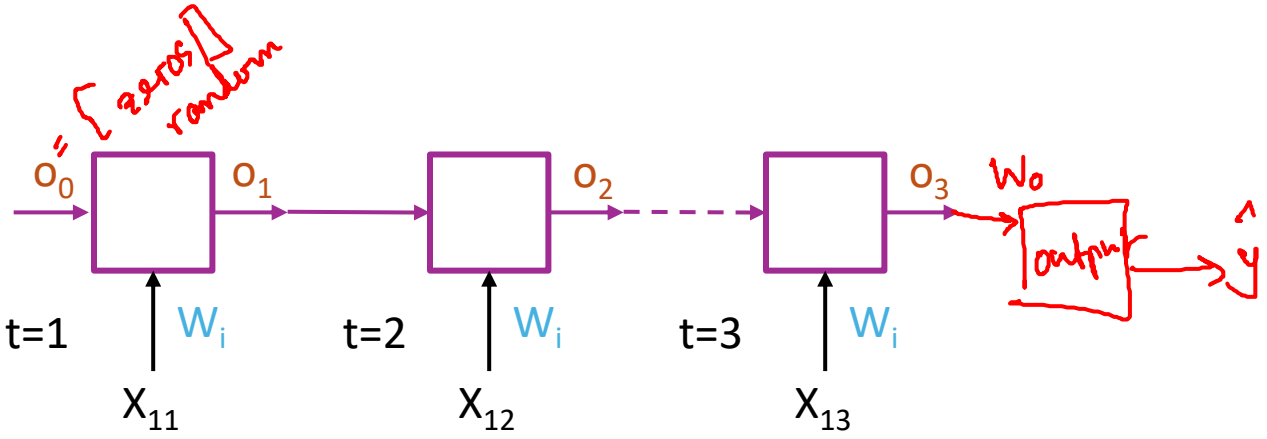


$$o_1 = f(X_{11}W_i + o_0W_h + b)$$

$$o_2 = f(X_{12}W_i + o_1W_h + b)$$

$$o_3 = f(X_{13}W_i + o_2W_h + b)$$

$$\hat{y} = \sigma(o_3W_o + b)$$



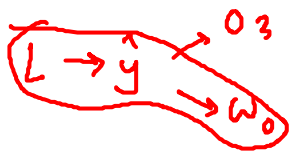
$$L = -y_i \log \hat{y}_i - (1 - y_i) \log (1 - \hat{y}_i)$$

gradient descent:

$$w_i = w_i - \eta \frac{\partial L}{\partial w_i}$$

$$w_h = w_h - \eta \frac{\partial L}{\partial w_h}$$

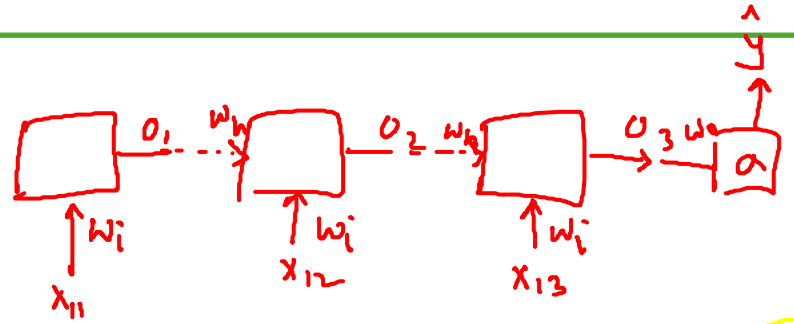
$$w_o = w_o - \eta \frac{\partial L}{\partial w_o}$$



$g(z)$

$g(z)(1-g(z))$

$\hat{y} = \sigma(o_3 w_0 + b) = g(z)$

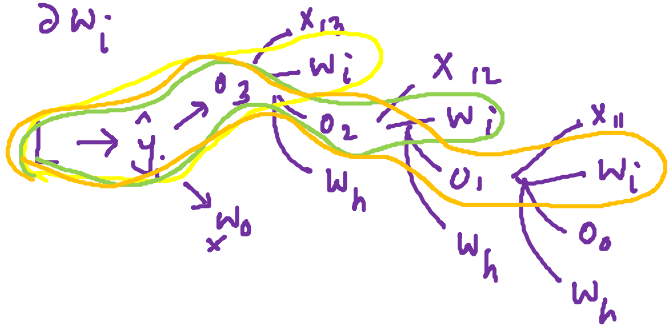


$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_0}$$

o_3

$g(z)(1-g(z)) \cdot o_3$

$\frac{\partial L}{\partial w_i}$

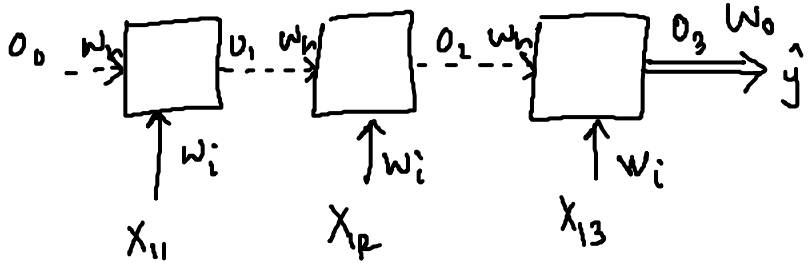


$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_i} + \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_i} +$$

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

$$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_i}$$

$$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_i}$$



$$\frac{\partial L}{\partial w_h} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_h} + \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_h} + \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_h}$$

$$\frac{\partial L}{\partial w_h} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_h}$$

Problem/Limitations of RNN. Vanilla RNN

① Long term dependency (lack of)

I did my MS in Germany, and that's how I, learn German.

~max 10 steps

② Unstable Gradients.

→ Vanishing gradients
- relu/leaky relu
- better weight initialization
- LSTMs

→ Exploding gradients
- Gradient clipping
- learning rate control
- LSTM