Bi-LSTMs (Bidirectional LSTMs)

1️⃣ Understanding LSTMs: Vanishing Gradient & Long-Term Dependencies

✅ Vanishing Gradient Problem

- In traditional **RNNs**, during backpropagation, the gradient values shrink as they are propagated backward.
- This leads to **very small weight updates**, making it hard to learn long-range dependencies.
- The deeper the network, the worse this problem gets.

✅ Long-Term Dependencies & How LSTMs Solve It

- **LSTMs (Long Short-Term Memory Networks)** are designed to handle long-term dependencies using **gates**:
  - **Forget Gate:** Decides what information to discard.
  - **Input Gate:** Decides what new information to store.
  - **Output Gate:** Decides what part of the memory cell to output.
- This **selective memory mechanism** helps LSTMs remember relevant past information over long sequences.

2️⃣ Forward & Backward Pass in Bi-LSTM

✅ What is Bi-LSTM?

- A **Bidirectional LSTM** processes the sequence **in both directions** (forward and backward).
- This allows the network to have **context from both past and future words**, unlike a standard LSTM that only looks backward.

✅ Forward Pass (Left to Right)

1. Process input from **t=1 to t=n** (left to right).
2. Store hidden states → in forward LSTM.

✅ Backward Pass (Right to Left)

1. Process input from **t=n to t=1** (right to left).
2. Store hidden states ← in backward LSTM.

✅ Final Output of Bi-LSTM

- The output at each time step is a **concatenation** of the hidden states from both directions:

$$ht = \text{concat}(ht\rightarrow, ht\leftarrow)$$

3️⃣ How Bi-LSTMs Improve Over LSTMs

✅ Key Benefits:

1. **Better Context Awareness**: LSTMs only see past data, while Bi-LSTMs see both past and future.
2. **Improved Performance**: Especially useful in NLP tasks like **Named Entity Recognition (NER)** and **Machine Translation**.
3. **Captures Dependencies**: Useful when context depends on future

words (e.g., in **question-answering systems**).

✅ **When NOT to Use Bi-LSTMs:**

- When **real-time processing** is required (e.g., speech recognition), as Bi-LSTM requires future words.

4️⃣ Implementing Bi-LSTMs Using PyTorch

Refer py notebook

5️⃣ Comparing Bi-LSTMs with Traditional LSTMs

Feature	LSTM	Bi-LSTM
Direction of Processing	One-direction (past to future)	Both directions (past & future)
Context Awareness	Limited (only previous words)	High (uses both past & future words)
Performance in NLP	Good	Better for context-dependent tasks
Computational Cost	Lower	Higher (double LSTM computations)
Use Case	Simple text processing	Tasks needing full context (NER, MT, Summarization)

🔥 Summary

- **LSTMs** solve the **vanishing gradient problem** but only process sequences **in one direction**.
- **Bi-LSTMs** process data **in both directions**, making them more powerful for **context-dependent NLP tasks**.
- **Implementation** is simple with **Keras (Bidirectional(LSTM))** and **PyTorch (bidirectional=True)**.
- **Better for NLP tasks** like **NER, Sentiment Analysis, Summarization, and Translation** but computationally expensive.