LSTM - Long Short-Term Memory

Understanding Through a Complete Example

Watch LSTM Process a Sentence

Sentence: "The cat was hungry. The dog was sleeping."

Word	Forget	Input	Output	Memory
"The"	0.9	0.3	0.2	[article]
"cat"	0.8	0.9	0.8	[cat]
"was"	0.9	0.7	0.9	[cat, was]
"hungry"	0.8	0.8	0.7	[cat, hungry]
"."	**0.1**	0.4	0.3	[end]
"The"	0.1	0.8	0.2	[article]
"dog"	0.7	**0.9**	0.9	[dog]
"was"	0.9	0.8	**0.9**	[dog, was]

Notice:

- Three mysterious numbers
- Memory changes
- "cat" disappears
- "dog" appears

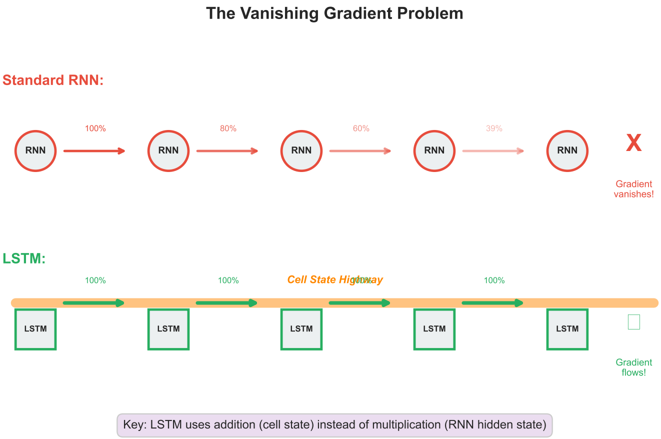
Intuition: The Magic

How does LSTM know to:

- Forget "cat"?
- Remember "dog"?
- Keep right info?

Let's find out...

Why Do We Need This?



RNN Problem:

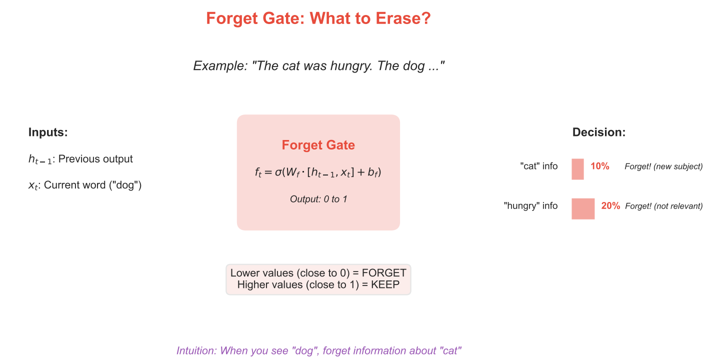
- Gradients vanish
- Forgets early info
- Can't handle long deps
- Loses "cat"

LSTM Solution:

- Cell state highway
- Addition not multiply
- Three gates
- Preserves then clears

RNN forgets "cat" by "dog"

Gate 1: Forget - What to Erase?



Remember Our Table?
Row 5: "." had **Forget = 0.1**

What This Means:

- 0.0 = forget everything
- 1.0 = keep everything
- 0.1 = forget most things

Why at period?

- New sentence starting
- Old subject no longer relevant

Formula:

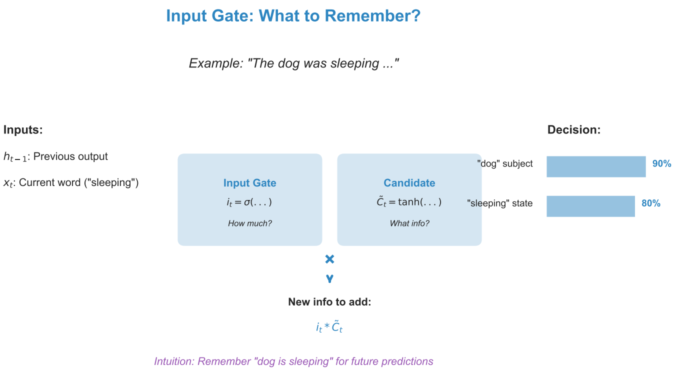
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

How It Decides:

1. Look at current word (".")
2. Look at previous output
3. Compute: Should we forget?
4. Output value 0 to 1

Cell state update:

Gate 2: Input - What to Add?



Remember Our Table?
Row 7: "dog" had **Input = 0.9**

What This Means:

- 0.0 = add nothing
- 1.0 = add everything
- 0.9 = add most of it

Why at "dog"?

- New subject appearing
- Important to remember

Formulas (Two Parts):

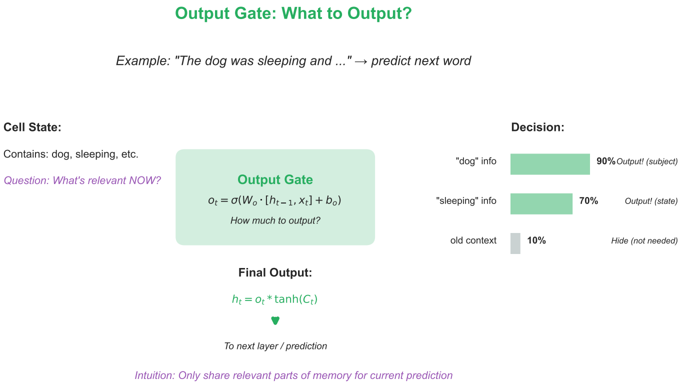
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

How It Works:

1. Create candidate info ($\tilde{C}_t$)
2. Decide how much to use (i_t)
3. Multiply them together
4. Add to cell state

Gate 3: Output - What to Share?



Remember Our Table?
Row 8: "was" had **Output = 0.9**

What This Means:

- 0.0 = output nothing
- 1.0 = output everything
- 0.9 = output most of memory

Why at "was"?

- Need to predict next word
- Subject info is relevant

Formulas:

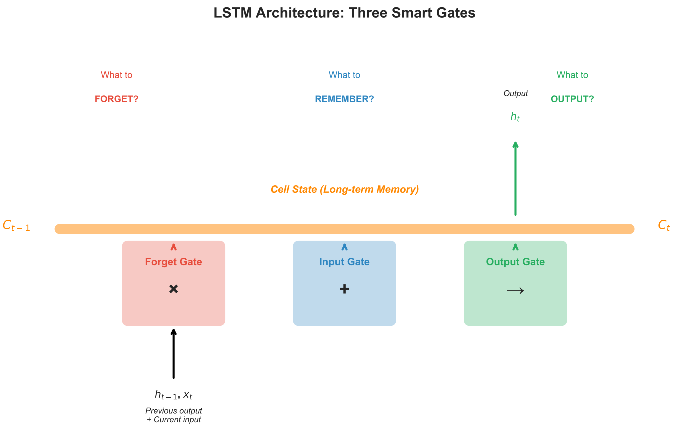
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

How It Works:

1. Look at cell state (has "dog")
2. Decide what's relevant now
3. Filter memory through gate
4. Send h_t to next layer

The Big Picture: Three Gates + Cell State



Cell State Highway:

- Protected memory
- Info flows easily
- Gates control entry/exit
- Gradients don't vanish!

All Three Work Together:
At each word:

1. **Forget:** Remove old (0.1 at "." = erase "cat")
2. **Input:** Add new (0.9 at "dog" = add subject)
3. **Output:** Share (0.9 at "was" = use "dog")

Result: Memory evolves!

Now Let's Look Again - You Understand It!

Sentence: "The cat was hungry. The dog was sleeping."

Word	Forget	Input	Output	Memory
"The"	0.9 (keep)	0.3 (small add)	0.2 (hide)	[article]
"cat"	0.8 (keep)	0.9 (add subject!)	0.8 (show)	[cat]
"was"	0.9 (keep)	0.7 (add verb)	0.9 (need it!)	[cat, was]
"hungry"	0.8 (keep)	0.8 (add state)	0.7 (show)	[cat, hungry]
"."	**0.1 (ERASE!)**	0.4 (ending)	0.3 (hide)	[end]
"The"	0.1 (clear old)	0.8 (new start)	0.2 (hide)	[article]
"dog"	0.7 (keep)	**0.9 (NEW subject!)**	0.9 (show)	[dog]
"was"	0.9 (keep)	0.8 (add verb)	**0.9 (USE dog!)**	[dog, was]

Checkpoint: The Critical Transition

Watch rows 4-7: "hungry" → "." → "The" → "dog"
Memory evolves: [cat, hungry] → FORGET → [end] → ADD → [dog]
This is what RNNs cannot do! LSTM uses gates to control memory intelligently.

The Complete Mathematics

All Three Gates:

Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell State Update:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Output:

$$h_t = o_t \odot \tanh(C_t)$$

In Our Example:

- x_t = current word embedding
- h_{t-1} = previous output
- C_t = cell state (memory)
- σ = sigmoid (0 to 1)
- tanh = tanh (-1 to 1)
- $\odot$ = element-wise multiply

Summary: LSTM in Practice

What We Learned:

1. LSTM uses three gates to control memory
2. Forget gate: what to erase (0.1 at "." = erase "cat")
3. Input gate: what to add (0.9 at "dog" = add subject)
4. Output gate: what to use (0.9 at "was" = use "dog")
5. Cell state flows information forward

When to Use LSTM:

- Long sequences (100+ words)
- Long-term dependencies
- Context matters
- Grammar and structure

Real World: Applications

Where LSTMs Excel:

- Machine Translation (Google Translate)
- Speech Recognition (Siri, Alexa)
- Text Generation (ChatGPT foundations)
- Video Analysis
- Music Generation
- Handwriting Recognition

Key Takeaway:
The sentence example showed you *exactly* how LSTM gates work in practice. That's the real magic!

Questions?