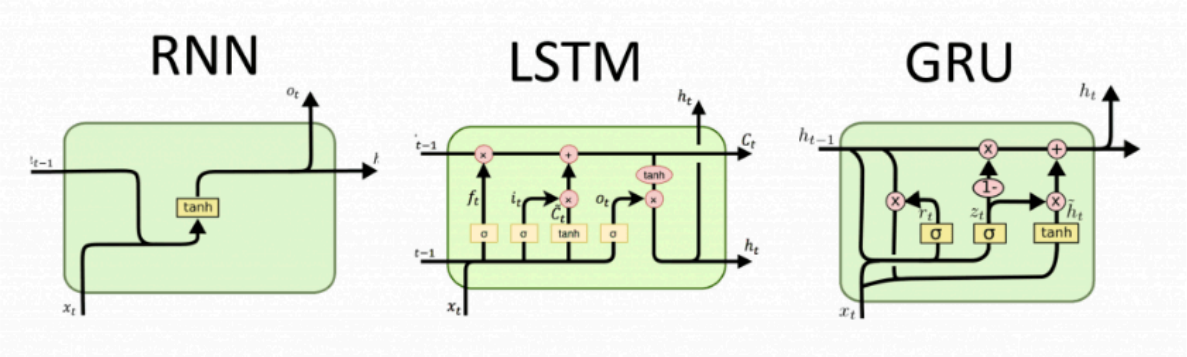
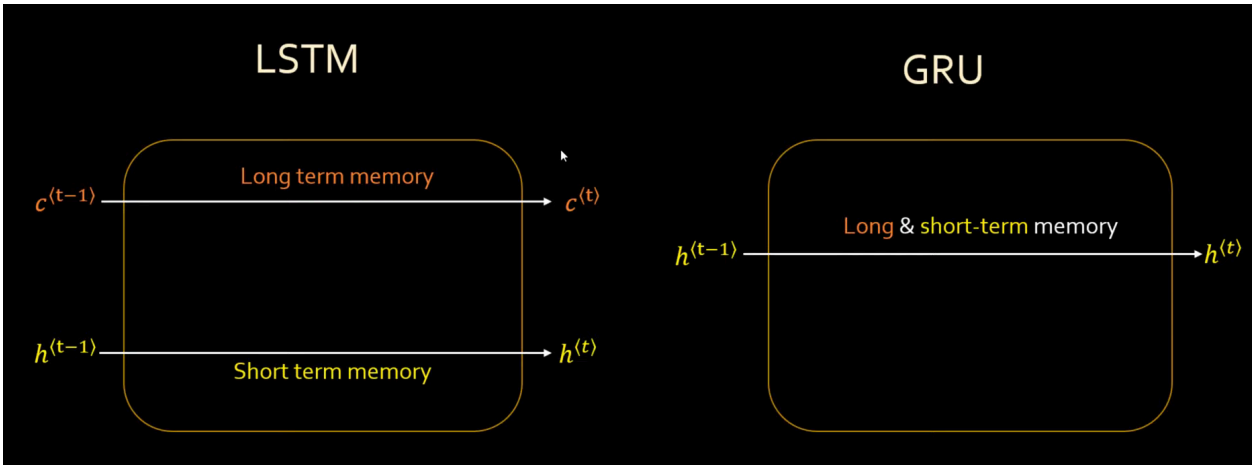
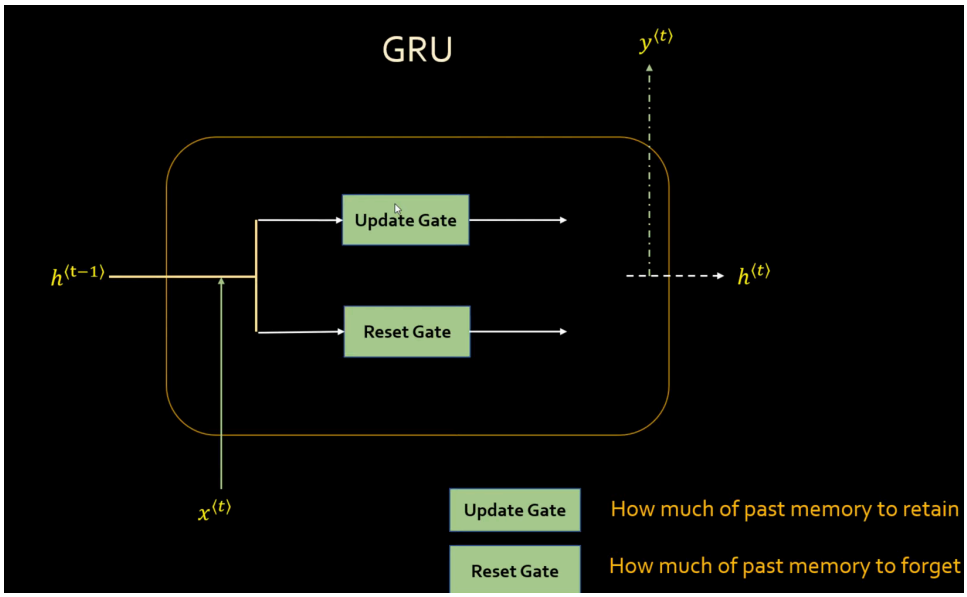
Gated Recurrent Unit (GRU)

- Simpler but powerful alternative to LSTM





🔷 Simple Summary



- **GRU** is a type of **Recurrent Neural Network (RNN)** architecture.
- Like LSTM, it's designed to **solve the vanishing gradient problem** and **learn long-term dependencies**.
- It's **simpler** than LSTM because it has **fewer gates** and **no separate memory cell**.

✅ It's **faster to train** and often works **just as well** as LSTM on many tasks.

🔶 Why Was GRU Introduced?

Problem with RNN:

- RNNs forget long-term information.
- They're hard to train due to **vanishing and exploding gradients**.

LSTM tried to fix this:

- It introduced a **memory cell** and **3 gates** (forget, input, output).
- But LSTM is **computationally heavy**.

So GRU was designed:

- To **keep the good parts of LSTM** (like controlling memory with gates)

- But with **less complexity**
- GRU uses only **2 gates**:
  - **Update Gate** (combines LSTM's **forget** + **input** gate)
  - **Reset Gate**

🎯 Input to the GRU

- x_t: input at current time step
- h_{t-1}: hidden state from the previous step
- (no separate c_t like LSTM; GRU merges cell state and hidden state)

How GRU Works? (The Simple Version)

Imagine GRU is a person reading a book:

1. **Update Gate**: Decides what new information to remember ("***This chapter is important, I'll remember it***")
   - Like a combination of LSTM's input and forget gates
   - Controls how much of the new content matters
   - Combines LSTM's **forget** and **input gate** into one.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

- If z_t = 1 → Keep all past memory
- If z_t = 0 → Forget and use only the new info

2. **Reset Gate**: Decides what past information to ignore ("T***he previous chapter isn't relevant anymore***")
   - Filters out unimportant old information
   - Helps focus on what matters now

- Decides **how much of the past memory to ignore**.
- Helps the GRU focus on the **recent inputs**.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

3. **Current Memory**: Creates new potential memories ("***What's important in this chapter?***")
    - Temporary storage of new information
    - Combines with the update gate's decision
    - Creates a **new candidate state** using reset memory and current input.
    - This is like saying:

    > "Here's what the memory could be right now."

$$\hat{h}_t = tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$$

4. Final Hidden State (h_t)

- **Mixes old and new memory** using the update gate.
- If update gate (z_t) is close to 1 → keep old memory
- If z_t is close to 0 → use the new candidate

$$h_t = (1 - z_t) * \hat{h}_t + z_t * h_{t-1}$$

🔁 Summary of GRU Flow

Step	Description	Formula
1	Update Gate	$z_t = \sigma(Wz[h_{t-1}, x_t])$
2	Reset Gate	$r_t = \sigma(Wr[h_{t-1}, x_t])$
3	Candidate State	$\hat{h}_t = \tanh(W[r_t * h_{t-1}, x_t])$
4	Final Output (hidden state)	$h_t = (1 - z_t) * \hat{h}_t + z_t * h_{t-1}$

🧪 Keras GRU Example

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense,Input
import numpy as np

# Step 1: Create model
model = Sequential()

model.add(Embedding(input_dim=10000, output_dim=64, input_length=100))

# Step 2: Add GRU layer
model.add(GRU(64))

# Step 3: Output layer
model.add(Dense(1, activation='sigmoid')
```

`input_dim` → How many unique words we have

`output_dim` → Size of each word vector

✅ GRU vs LSTM — Comparison

Feature	LSTM	GRU
Gates	3 (input, forget, output)	2 (reset, update)
Memory cell	Yes (C_t)	No (merged with hidden state)
Complexity	Higher	Lower
Training Speed	Slower	Faster
Accuracy	Similar in most tasks	Similar or better
Best use cases	Long sequences (e.g., language)	Short/medium sequences (e.g., time series)

Use GRU when?

- You want faster training
- Your sequences aren't extremely long
- You want simpler model architecture

Use LSTM when:

- Working with very long sequences
- Need maximum performance (may be slightly better)
- Don't mind extra computation

Real-World Example

For a movie review classifier:

- GRU would read the review and decide:
    1. Which words are important (update gate)
    2. Which earlier words to ignore (reset gate)
    3. Combine this to understand the overall sentiment

Common Applications

1. Text classification
2. Speech recognition
3. Time series forecasting
4. Machine translation
5. Any sequence modeling task