**Daily Learning Report: Understanding LSTMs for Time Series Prediction**

**Date**: [Insert Date]
**Author**: [Your Name]

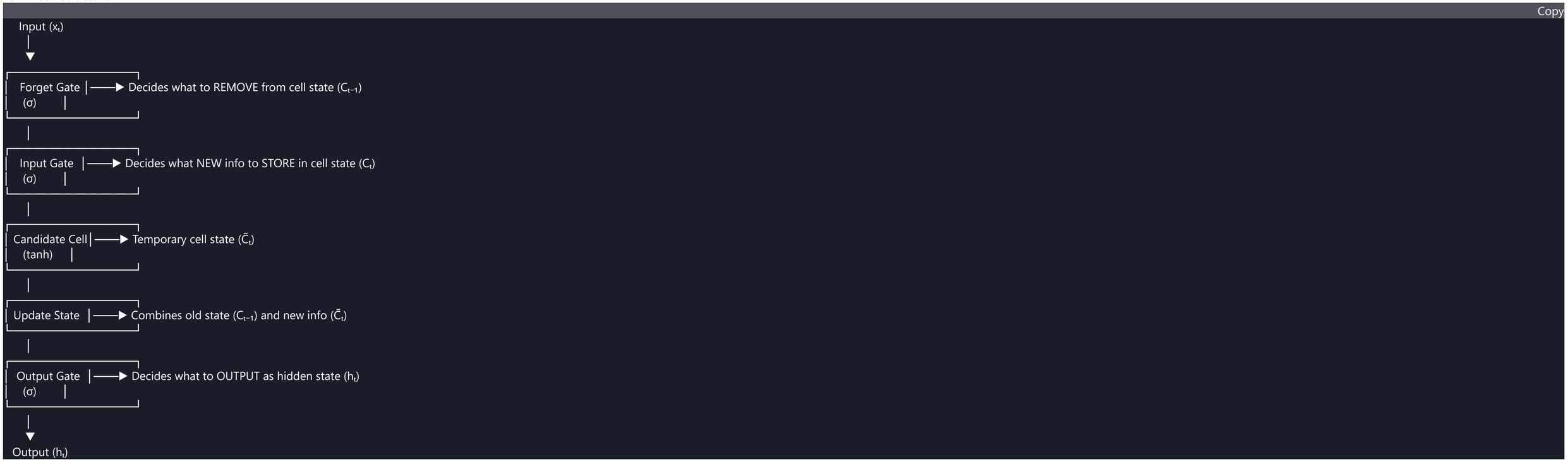
**1. Introduction**

Today's focus was on understanding **Long Short-Term Memory (LSTM)** networks, a type of Recurrent Neural Network (RNN) designed to handle sequential data like stock prices, weather data, or text. Below is a summary of key concepts, diagrams, and takeaways.

**2. What is an LSTM?**

An LSTM is a neural network with **memory cells** that can retain information over long periods. It uses **gates** to control what information to keep, forget, or pass to the next timestep.

**LSTM Cell Structure**:




*Key*:

**σ**: Sigmoid (squishes values to 0–1).

**tanh**: Hyperbolic tangent (squishes values to -1–1).

**3. How LSTMs Process Sequences**

For time series prediction, LSTMs process sequences using **sliding windows**.

**Example: Predicting Day 4 from Days 1–3**


**Hidden State ($h_t$)**: Passed to the next timestep (short-term memory).

**Cell State ($C_t$)**: Carries long-term dependencies (e.g., trends).

**4. Key Concept: Parameter Sharing**

All LSTM cells in a layer **share the same weights and biases** ( `W_f` , `W_i` , `W_C` , `W_o` , `b_f` , etc.).

**Diagram: Shared Weights Across Windows**


**Final Weights ($W_n$, $b_n$)**: Used to predict test data (e.g., Days 8–10 → Day11).

**5. Training Process**

**Step 1: Forward Pass**
Process the sequence (e.g., Days 1–3) to predict the next value (Day4).
Compute loss (e.g., Mean Squared Error).
**Step 2: Backward Pass (BPTT)**
Calculate gradients for shared weights using **Backpropagation Through Time (BPTT)**.
Update weights to minimize loss.
**Step 3: Repeat for All Windows**
Train on all windows to refine weights iteratively.
**Diagram: Gradient Accumulation**

```
Window 1 Gradients →
Window 2 Gradients → ├──► Aggregate Gradients → Update W, b
Window 3 Gradients →
```

**6. Testing the Model**
Use the **final weights ($W_n$, $b_n$)** from training to predict unseen sequences.
Example: Predict Day11 using Days 8–10.
**Test Data Flow**

```
[Day8] → [Day9] → [Day10] → Predict Day11
  ▼       ▼       ▼        ▼
LSTM → LSTM → LSTM → Dense → Prediction (ŷ)
```

**7. Common Misconceptions**

| Myth | Fact |
| --- | --- |
| Each window has separate weights. | All windows share the same weights ( `W, b` ). |
| Training on one window is enough. | Train on all windows to generalize. |
| Hidden states carry across windows. | Hidden states reset for each new window. |

**8. Key Takeaways**
. **Parameter Sharing**: LSTMs reuse weights across all timesteps/windows.
. **Gates Control Memory**: Forget/Input/Output gates manage information flow.
. **Training**: Updates weights iteratively using gradients from all windows.
. **Prediction**: Final weights ( $W_n$, $b_n$ ) predict unseen sequences.
**9. Next Steps**
**Code Implementation**: Try building an LSTM with TensorFlow/PyTorch.
**Hyperparameter Tuning**: Experiment with `look_back` window size, hidden units.
**Advanced Topics**: Bidirectional LSTMs, Attention Mechanisms.

**End of Report**
Let me know if you'd like a code example or further details! 😊