Deep Learning for Sequential Data: Understanding RNNs and LSTMs Intuitively

Week 8: FinTech Course
Intuitive Version

November 5, 2025

Abstract

This document provides an intuitive, conceptual understanding of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. Rather than mathematical formalism, we focus on analogies, visual thinking, and practical understanding. You'll learn how these networks "remember" past information, why regular RNNs struggle with long-term memory, and how LSTMs elegantly solve this problem. Perfect for building intuition before diving into the mathematics!

1 The Problem: Why Do We Need Special Networks?

1.1 Traditional Neural Networks Are Forgetful

Imagine you're trying to predict tomorrow's stock price. A traditional neural network would look at today's price in isolation:

"Today's price is \$100" → *Network* → *"Tomorrow will be \$102"*

But this is like trying to predict the weather by only looking out your window **right now**, ignoring that it's been raining all week!

The Core Problem

Financial data has **memory**. What happened yesterday influences today. What happened last week influences this week. Traditional neural networks have **amnesia** – they process each data point independently, forgetting everything that came before.

1.2 What Makes Financial Data Sequential?

Think about Bitcoin's price:

- **Momentum:** If Bitcoin rose yesterday, it's more likely to rise today
- **Trends:** Bull markets last weeks or months
- **Volatility clustering:** Crazy price swings cluster together
- **Events:** A major news event affects prices for days
- **Cycles:** Weekly patterns (weekday vs weekend trading)

The Story Analogy

Reading a book one word at a time vs. reading random words:

- **With sequence:** "The quick brown fox jumps over the lazy dog"
- **Without sequence:** "Dog the over fox jumps lazy brown quick the"

The same words, but only the first makes sense! Financial data is like a story – order matters.

1.3 The Order of Data Matters

Consider these two price sequences:

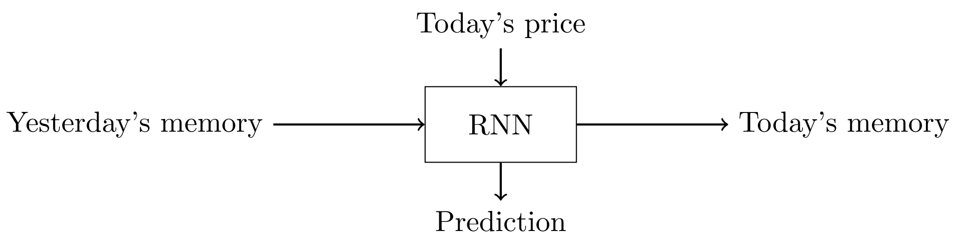
- **Sequence A:** \$100 → \$110 → \$105 → ?
- **Sequence B:** \$105 → \$100 → \$110 → ?

Same three prices, different order, completely different predictions! We need networks that understand sequences.

2 Enter Recurrent Neural Networks (RNNs)

2.1 The Big Idea: Networks with Memory

An RNN is like a neural network with a **short-term memory**. It processes data one step at a time, keeping a "mental note" of what it's seen before.



RNN's Memory Mechanism

At each time step, an RNN:

1. Takes current input (today's data)
2. Recalls its previous memory (what it learned from yesterday)
3. **Updates its memory** by combining both
4. Makes a prediction
5. Passes the updated memory to tomorrow

2.2 The Conversation Analogy

Think of an RNN as having a conversation with you:

RNN as a Conversational Partner

You: "Bitcoin hit $50,000 today"
RNN: *remembers: high price*

You: "It's up 10% from yesterday"
RNN: *updates memory: high price + strong momentum*

You: "Trading volume doubled"
RNN: *updates memory: high price + momentum + high interest*

You: "What's tomorrow's price?"
RNN: "Based on everything we discussed, probably $52,000"

The RNN doesn't just hear your last sentence – it remembers the whole conversation!

2.3 How RNNs Process Sequences

Let's trace through predicting Bitcoin price on Wednesday:

Monday:

- Input: $48,000

- Memory: (empty – first day)
- Update memory: “Started at $48K”
- Prediction: $48,500

Tuesday:

- Input: $49,000
- Memory: “Started at $48K”
- Update memory: “Started $48K, rising trend”
- Prediction: $50,000

Wednesday (prediction day):

- Input: $50,000
- Memory: “Started $48K, rising trend”
- Update memory: “Started $48K, strong uptrend”
- **Prediction: $51,500** (accounting for the trend!)

2.4 The Snowball Effect

Think of RNN memory like a snowball rolling down a hill:

- Starts small (initial memory)
- Picks up snow at each step (new information)
- Gets bigger and bigger (accumulating context)
- Carries everything from the past (complete history)

3 The Problem with Basic RNNs

3.1 The Fading Memory Problem

Here’s the catch: RNNs have **short-term memory loss**. Like a game of telephone, information gets distorted the longer it travels.

The Telephone Game

Remember playing telephone as a kid?

1. First person: “The quick brown fox”
2. After 5 people: “The thick clown box”
3. After 10 people: “Something about foxes?”
4. After 20 people: “Uh... what were we talking about?”

Basic RNNs have the same problem. After 50-100 time steps, they’ve essentially forgotten what happened at the beginning.

3.2 Why Does This Happen?

Imagine passing a note through 100 people, and each person:

- Reads the note
- Rewrites it in their own words
- Adds today's news
- Passes it on

By person #100, the original message is completely lost! This is the **vanishing gradient problem**.

The Vanishing Gradient Problem

When training RNNs, we need to send feedback backward through time. But this feedback signal gets weaker and weaker ("vanishes") as it travels back. It's like shouting through 100 rooms – by the last room, nobody can hear you!
Result: RNNs can't learn patterns that span more than 10-20 time steps.

3.3 Real-World Impact

For Bitcoin prediction, this means:

- Can learn: "Yesterday's spike affects today"
- Can learn: "This week's trend continues"
- × Cannot learn: "The halving 6 months ago drives current prices"
- × Cannot learn: "Institutional adoption over the year matters"

3.4 The Conversational Memory Limitation

Going back to our conversation analogy:

Short-Term Memory in Conversations

You: "I'm planning a trip to Japan"
RNN: *remembers*
... 100 sentences later ...
You: "So do you remember where I'm traveling?"
RNN: "Uh... you mentioned travel... somewhere in Asia? Was it China? I forget!"

Basic RNNs are like someone with poor long-term memory – they remember the last few things perfectly but forget everything from 10 minutes ago.

4 Long Short-Term Memory (LSTM): The Solution

4.1 The Revolutionary Idea

LSTMs are like RNNs with a **notebook**. Instead of trying to remember everything in their head, they:

1. Keep a notebook for important long-term information
2. Use short-term memory for immediate context
3. Decide what to write down, what to forget, and what to share

LSTM's Two-Memory System

LSTMs maintain TWO types of memory:

- **Cell State (Long-term memory):** Like a notebook that persists
- **Hidden State (Working memory):** Like short-term memory for immediate use

4.2 The Executive Assistant Analogy

Think of an LSTM as an executive with a smart assistant:

LSTM as Executive + Assistant

The Executive (Hidden State):

- Handles day-to-day decisions
- Processes current information
- Makes predictions

The Assistant (Cell State):

- Maintains a detailed log book
- Stores important information long-term
- Retrieves relevant context when needed

The Magic: Three smart interns (gates) decide:

1. What to forget from the log book
2. What new information to write down
3. What to tell the executive right now

4.3 The Three Gates: LSTM's Smart Filters

LSTMs use three "gates" – think of them as smart filters or decision-makers:

4.3.1 1. The Forget Gate: What to Forget

The Forget Gate in Action

Situation: Bitcoin crashed from \$60K to \$30K. News: "Bull market over!"
Forget Gate Decision:

- Old memory: "We're in a bull market" → FORGET THIS
- Keep: "Bitcoin is volatile" → KEEP THIS
- Keep: "Institutional interest exists" → KEEP THIS

The forget gate looks at current situation and decides what old information is no longer relevant.

Analogy: Like cleaning out your notebook – some notes from last year are still useful, others are outdated.

4.3.2 2. The Input Gate: What to Remember

The Input Gate in Action

New Information: "Major exchange got hacked, \$500M stolen"
Input Gate Decision:

- "Exchange hacked" → VERY IMPORTANT! Write this down!
- "Minor price fluctuation" → Less important, don't write down
- "Random tweet" → Ignore, not worth remembering

The input gate decides what new information is important enough to store in long-term memory.

Analogy: Like deciding what to write in your diary – major life events get an entry, routine stuff doesn't.

4.3.3 3. The Output Gate: What to Share Now

The Output Gate in Action

Question: "Should I buy Bitcoin today?"
Output Gate looks at the notebook and decides what's relevant:

- Recent hack news → SHARE: Affects short-term price
- Long-term adoption trend → Mention: Affects long-term
- Historical halving from 2020 → Don't share: Not relevant now

The output gate filters what information from long-term memory is relevant for the current decision.

Analogy: Like a lawyer reviewing case files – bring up relevant precedents, ignore irrelevant ones.

4.4 How LSTMs Process Information: A Story

Let's follow a Bitcoin prediction through an LSTM:

Day 1: Bitcoin halving event

1. *Input:* "Halving occurred, supply cut in half"
2. *Input Gate:* "This is HUGE! Write this down in the notebook!"
3. *Forget Gate:* "Nothing to forget yet"
4. *Output Gate:* "Share this with prediction"
5. **Cell State (notebook):** "Halving just happened"
6. **Prediction:** "Expect slow price rise"

Day 30: Small daily fluctuation

1. *Input:* "Price up 0.5% today"
2. *Input Gate:* "Minor movement, don't write this down"
3. *Forget Gate:* "Keep halving info, it's still relevant"
4. *Output Gate:* "Mention the halving context"
5. **Cell State (notebook):** "Halving happened 30 days ago" (still there!)
6. **Prediction:** "Continue gradual rise"

Day 180: Strong momentum builds

1. *Input:* "Price up 50% in one month!"
2. *Input Gate:* "This is important! Write it down!"
3. *Forget Gate:* "Keep halving info, it explains this"
4. *Output Gate:* "Share both halving effect and momentum"
5. **Cell State (notebook):** "Halving 6 months ago + strong momentum"
6. **Prediction:** "Bull market confirmed, expect continuation"

Notice how the halving information persisted for 180 days!

4.5 Why LSTMs Don't Forget

The secret is in how information flows:

The Information Highway

The cell state is like a **protected highway** for information:

- Information can flow unchanged across hundreds of time steps
- Gates make small additions or deletions
- No forced "rewriting" at each step (unlike vanilla RNNs)
- Think: passing a baton vs. rewriting a message

Vanilla RNN: Rewrites entire memory each step
LSTM: Keeps notebook intact, makes small edits

4.6 The Notebook Analogy Revisited

LSTM's Notebook System

Imagine maintaining a trading journal:
Vanilla RNN:

1. Read yesterday's journal entry
2. Completely rewrite it with today's info
3. After 100 days, original insight is lost

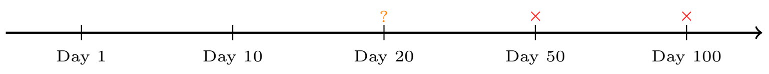
LSTM:

1. Keep a running log that persists
2. Add new important insights
3. Cross out outdated information
4. Highlight relevant parts for today
5. Original insights from Day 1 can still exist on Day 100!

5 Comparing RNN and LSTM

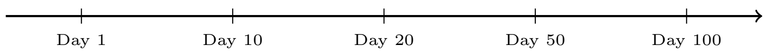
5.1 Visual Comparison

Vanilla RNN Memory:



Memory strength fades quickly

LSTM Memory:



Memory preserved indefinitely

5.2 Capability Comparison

Capability	**RNN**	**LSTM**
Short-term patterns (1-10 days)		
Medium-term patterns (10-50 days)	∼	
Long-term patterns (50+ days)	×	
Training speed	Fast	Slower
Number of parameters	Fewer	More
Memory requirements	Lower	Higher
Prevents vanishing gradients	×	

5.3 When to Use Which

Decision Guide

Use Vanilla RNN when:

- Patterns are short-term (intraday trading)
- You have limited computational resources
- Data has high sampling frequency (minute-by-minute)
- Interpretability is crucial

Use LSTM when:

- You need long-term memory (swing trading, position trading)
- Patterns span weeks or months
- You have sufficient training data
- Performance is more important than speed

6 Real-World Financial Applications

6.1 Price Prediction

The Task: Predict Bitcoin price tomorrow based on last 30 days

LSTM's Advantage

What LSTM Remembers:

- Recent trend (last week)
- Major support/resistance levels (last month)
- Important events (exchange hack 2 weeks ago)
- Seasonal patterns (end of month rally)
- Long-term bull/bear context

What Vanilla RNN Forgets:

- $\times$ Events from 2 weeks ago
- $\times$ Support levels from early in the month
- $\times$ The beginning of the current trend

6.2 Volatility Forecasting

The Pattern: Volatility clusters – high volatility days come in groups

Weather Analogy

Volatility clustering is like weather patterns:

- One rainy day often means more rain tomorrow
- Storm systems last several days
- Eventually, calm weather returns

LSTM advantage: Can remember "we entered a high-volatility regime 2 weeks ago" and predict continuation, even after a few calm days.

6.3 Sentiment Analysis

The Task: Analyze news articles about a company over time

Sequential Sentiment

Week 1: "Company launches new product" → Positive
Week 2: "Product receives mixed reviews" → Neutral
Week 3: "Sales disappoint" → Negative
Week 4: "CEO announces turnaround plan" → ?
LSTM Context: Remembers the product launch, mixed reception, and sales disappointment. This context makes the turnaround plan more significant (desperate move) vs. routine announcement.
RNN Context: Only remembers last week's sales disappointment, misses the full narrative arc.

6.4 Portfolio Rebalancing

The Decision: When to rebalance your portfolio

Multi-Asset Context

LSTM tracks multiple threads simultaneously:

- Bitcoin trend: Strong uptrend for 3 months
- Ethereum: Consolidating for 2 weeks
- Traditional stocks: Bear market for 6 months
- Bond yields: Rising steadily for 4 months
- Dollar strength: Weakening for 8 months

Decision: Combine all these long-term contexts to decide rebalancing timing. RNNs would struggle to maintain all these parallel threads in memory.

7 Training LSTMs: A Conceptual View

7.1 How LSTMs Learn

Think of training as teaching the LSTM to be a good note-taker:

Training as Learning to Take Notes

Untrained LSTM: Like a student who writes down everything or nothing

- Forget gate: Randomly forgets important things
- Input gate: Writes down irrelevant details
- Output gate: Shares random information

After training: Like an experienced analyst

- Forget gate: "This news is 3 months old and resolved, forget it"
- Input gate: "Major regulatory change – definitely write this down!"
- Output gate: "For today's prediction, the recent trend matters most"

7.2 The Training Process

1. **Make predictions:** LSTM predicts prices for historical data
2. **Compare to reality:** How wrong were the predictions?
3. **Assign blame:** Which gate made bad decisions?
4. **Adjust gates:** Make gates smarter
5. **Repeat:** Do this thousands of times

What Changes During Training

The LSTM learns:

- **What to forget:** "Day-to-day noise isn't important, but major events are"
- **What to remember:** "Halvings matter for months, minor news matters for days"
- **What to use:** "For short-term predictions, use recent data; for long-term, use context"
- **How to combine:** "Weight recent trend 70%, long-term context 30%"

7.3 Why Training Takes Longer Than RNNs

LSTM Training Complexity

RNN training: Like teaching someone to remember one thing
LSTM training: Like teaching someone to:

- Keep a detailed notebook
- Decide what's worth writing
- Organize information by importance
- Retrieve relevant context
- Update beliefs based on new information

Result: 4× more parameters to learn, 2-3× longer training time
But: Much better results for long-term patterns!

8 Common Pitfalls and Solutions

8.1 Pitfall 1: Not Enough Data

The Data Hunger Problem

Analogy: Learning to predict weather
With 10 days of data:

- "It rained yesterday, so it might rain today" (simple pattern)

With 10 years of data:

- "It's summer, we're in an El Niño year, pressure system from Pacific..."
- (complex seasonal patterns, multiple factors)

Rule of thumb: Need at least 1000 time steps, preferably 5000+ for good LSTM performance.

8.2 Pitfall 2: Overfitting the Past

The Memorization Problem

Bad student: Memorizes exam answers without understanding
Good student: Learns concepts that apply to new questions
Overfitted LSTM:

- "Bitcoin always drops 10% on Tuesdays in March" (memorized noise)
- Perfect on training data, terrible on new data

Well-trained LSTM:

- "High volatility tends to persist" (learned pattern)
- "Major news moves markets" (general principle)

Solution: Dropout (randomly "turn off" parts of the network during training, forcing it to learn robust patterns)

8.3 Pitfall 3: Wrong Time Scale

Choosing Look-Back Period

Too short (5 days):

- × Misses longer trends
- × Too reactive to noise
- ∼ Like making decisions based on this week only

Too long (200 days):

- × Too much irrelevant history
- × Slower to train
- ∼ Like considering your entire childhood for today's lunch decision

Just right (20-60 days):

- Captures medium-term trends
- Includes relevant context
- Not overwhelmed by noise

8.4 Pitfall 4: Unrealistic Expectations

What LSTMs Can and Cannot Do

LSTMs CAN:

- Learn patterns from data
- Capture long-term dependencies
- Improve predictions vs. simpler models
- Adapt to different regimes

LSTMs CANNOT:

- Predict completely random movements
- Foresee unprecedented events
- Guarantee profitable trading
- Replace fundamental analysis

Remember: Even the best LSTM can't predict lottery numbers! Markets have randomness that no model can capture.

9 Building Intuition: Thought Experiments

9.1 Experiment 1: The Halving Effect

Scenario: Bitcoin halving affects prices for 12-18 months after the event.

RNN vs LSTM Prediction

Month 1 after halving:

- RNN: "Halving just happened" – predicts well
- LSTM: "Halving just happened" – predicts well

Month 3 after halving:

- RNN: $\sim$ "Halving memory fading" – okay predictions
- LSTM: "Halving 3 months ago" – good predictions

Month 12 after halving:

- RNN: $\times$ "What halving?" – poor predictions
- LSTM: "Historical halving effect" – captures the pattern

9.2 Experiment 2: Multiple Time Scales

Scenario: Predict daily prices considering:

- Weekly momentum (7 days)

- Monthly support levels (30 days)
- Quarterly trends (90 days)
- Yearly cycles (365 days)

LSTM's Hierarchical Memory

LSTM can maintain multiple "threads" of information simultaneously:

- Recent thread: "Strong buy pressure this week"
- Medium thread: "Testing $50K resistance for a month"
- Long thread: "In Q4 bull market since January"
- Very long thread: "Post-halving year, historically bullish"

All these contexts inform today's prediction!

9.3 Experiment 3: Regime Changes

Scenario: Market shifts from bull to bear market

Adaptive Memory

Think of LSTM's memory as a **living document** that updates:
Bull Market (Months 1-6):

- Notebook says: "Bull market, buy dips, HODL"
- Predictions: Optimistic

Transition (Month 7):

- New info: "Major crash, -50% in one month"
- Forget gate: Cross out bull market notes
- Input gate: Write: "Bear market started"
- Notebook updates: "Entering bear market"

Bear Market (Months 8-12):

- Notebook says: "Bear market, caution"
- But also remembers: "Previous bull market peaked at $60K"
- Predictions: Cautious, with context of recent history

10 Practical Guidelines for Using LSTMs

10.1 Data Preparation

Getting Data Ready

1. **Normalization is Critical**
   - Scale prices to 0-1 range
   - *Why?* LSTM gates use sigmoid/tanh functions that work best in this range
   - *Analogy:* Like converting all currencies to dollars before comparing
2. **Sequence Length Matters**
   - Use 20-60 time steps for most financial data
   - *Why?* Balance between context and computational cost
   - *Analogy:* Reading last chapter vs. entire book for context
3. **Train/Val/Test Split**
   - Time-based split, not random!
   - *Why?* Can't use future to predict past
   - *Analogy:* Can't study tonight's exam after taking it

10.2 Model Architecture

Choosing LSTM Size

Number of LSTM Units:

- Too few (10-20): Underfitting, can't capture patterns
- Just right (50-100): Good balance
- Too many (500+): Overfitting, memorizes noise

Number of LSTM Layers:

- Single layer: Good for simple patterns
- 2 layers: Captures hierarchical patterns (short + long term)
- 3+ layers: Usually overkill for finance

Rule of thumb: Start simple (1 layer, 50 units), increase if needed.

10.3 Training Strategy

Training Best Practices

1. **Start with baseline:** Always compare to simple models
   - "Tomorrow = Today" (naive forecast)
   - Moving average
   - Linear regression
2. **Use early stopping:**
   - Monitor validation performance
   - Stop when it stops improving
   - *Why?* Prevents overfitting
3. **Be patient:**
   - LSTMs need many epochs (50-100+)
   - Progress can be slow
   - *Analogy:* Like learning a musical instrument – takes time!

11 Conclusion: The Big Picture

11.1 What We've Learned

1. **Sequences matter:** Financial data is ordered, not random
2. **RNNs add memory:** But short-term memory only
3. **LSTMs solve forgetting:** Through smart gates and dual memory
4. **Trade-offs exist:** Complexity vs. performance vs. interpretability
5. **Not magic:** Tools that work when used properly

11.2 The Core Insight

LSTM's Revolutionary Idea

Separate what to remember from what to use
Traditional thinking: Store everything in one place
LSTM thinking:

- Long-term storage (cell state)
- Working memory (hidden state)
- Smart filters (gates) between them

This separation enables learning from both recent context and distant past simultaneously.

11.3 When to Use LSTMs

Decision Framework

LSTM is worth it when:

- You have 1000+ time steps of data
- Patterns span 20+ time steps
- Simple models fail
- You can afford training time
- Performance improvement matters

Simpler models better when:

- • Limited data (¡ 500 observations)
- • Short-term patterns only (¡ 10 steps)
- • Need interpretability
- • Computational resources limited
- • Real-time predictions needed

11.4 Final Thoughts

LSTMs are powerful but not magical. They're tools that:

- Learn patterns from data
- Require lots of data and training time
- Excel at long-term dependencies
- Still can't predict unpredictable events

"The best model is the simplest one that works."

Start simple, increase complexity only when needed. Always compare to baselines. Understand why your model works, not just that it works.

Further Exploration

To deepen your intuition:

1. Implement a simple RNN from scratch (understand the basics)
2. Add gates one at a time (see how each improves performance)
3. Visualize gate activations (see what the model "decides")
4. Try different sequence lengths (feel the trade-offs)
5. Compare to simpler models (appreciate when complexity helps)

Remember: Intuition comes from practice, not just reading. Build, experiment, fail, learn!