SCALER School of Technology × Priyansh Saxena

Sequence Modeling: RNNs and the Temporal Dimension

Vanilla RNN • Hidden State Dynamics • BPTT • Vanishing Gradients

Priyansh Saxena

Scaler School of Technology | Deep Learning I: Neural Networks | Week 6 — Session 2/2

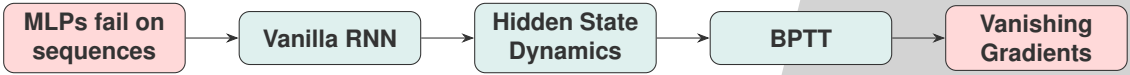
May 2026

The World is Sequential

- All of these have a **temporal/order dimension**
- Shuffling the elements destroys meaning
- CNNs assume spatial locality — but sequences are **temporal**

“CNNs crushed images. But what about data where ORDER matters?”

Today’s Roadmap

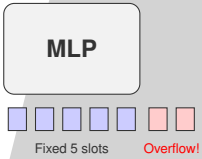


By the end: you’ll understand both the **power** and the **fundamental limitation** of RNNs.

The Problem with MLPs for Sequences

Three fatal problems:

1. **Fixed input size** — MLP needs a predetermined number of input neurons
2. **No notion of order** — position 1 and position 100 are unrelated features
3. **Parameter explosion** — 500 words $\times$ 300-dim = 150,000 input neurons!



The Issue

Flattening a sentence into one long vector loses all temporal structure.

What Makes Sequences Special

1. **Variable length** — sentences, songs, time series all differ in length
2. **Order matters** — “dog bites man” $\neq$ “man bites dog”
3. **Long-range dependencies** — meaning can depend on distant elements

Example: Long-Range Dependency

“The students who came from France _____ French.”

The answer (“speak”) depends on “students” — 6 words back!

The Key Insight: Weight Sharing Across Time

Solution

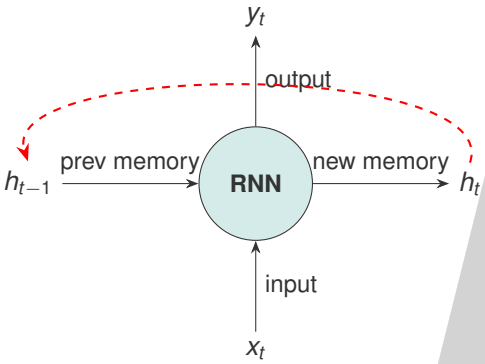
Apply the **same function** at every timestep.

- Same weights process word 1 and word 1000
- Handles **any length** — just keep applying
- **Constant parameters** regardless of sequence length

Analogy

"Your brain doesn't grow a new neuron for each word you read. The same neural circuits process every word."

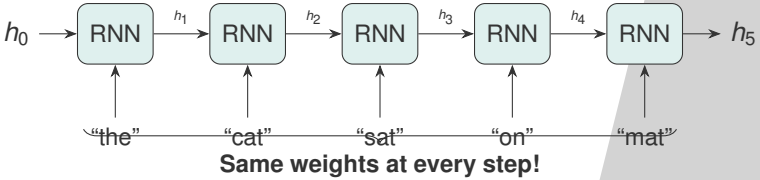
RNN: A Network with Memory



The Recurrence Equation

$$h_t = \tanh(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h)$$

Unrolling the RNN



- Folded view (loop) = compact notation
- Unrolled view (chain) = what actually happens during computation
- These are the **same network** copied across time

The Three Weight Matrices

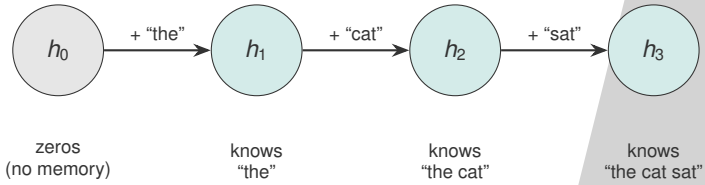
Matrix	Connection	Role
W_{xh}	Input $\rightarrow$ Hidden	How to process new input
W_{hh}	Hidden $\rightarrow$ Hidden	How to update memory
W_{hy}	Hidden $\rightarrow$ Output	How to produce predictions

Key Property

For a sequence of length T:

- MLP parameters: $O(T \cdot d^2)$ — grows with length!
- RNN parameters: $O(d^2)$ — **constant** regardless of length!

Step-by-Step: Processing “the cat sat”



- Start with $h_0 = \mathbf{0}$ (no prior memory)
- Each step: combine current word with accumulated context
- h_3 encodes the **entire phrase** in a single vector
- Adding more words? Same process, just keep going!

RNN in PyTorch

```python
import torch.nn as nn

# Create RNN layer
rnn = nn.RNN(input_size=50, hidden_size=128,
batch_first=True)

# Forward pass
x = torch.randn(1, 20, 50)  # (batch, seq_len, features)
output, h_n = rnn(x)        # output: all h_t, h_n: final h
```

- `output`: hidden states at **all** timesteps — shape `(1, 20, 128)`
- `h_n`: **final** hidden state only — shape `(1, 1, 128)`
- Same 3 weight matrices inside, regardless of sequence length

Quick Check

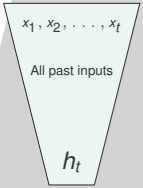
1. How many **unique** weight matrices in a vanilla RNN?
   - ✓ 3 — W_{xh}, W_{hh}, W_{hy} (shared across all timesteps)
2. What is the hidden state's job?
   - ✓ Compress all past information into a fixed-size vector
3. True or False: The RNN uses **different** weights at each timestep.
   - ✓ **False** — same weights everywhere (that's the whole point!)

Coming Up Next

Part 2: Hidden State Dynamics, BPTT, and Vanishing Gradients

The Hidden State as Compressed Memory

- h_t summarizes **everything** from x_1 to x_t
- It's a **lossy compression** — can't reconstruct all past inputs
- The network **learns** what to remember during training



Analogy

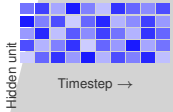
"Reading a 300-page novel. At page 150, you have a mental model of plot, characters, tension. That mental model = hidden state."

What Gets Encoded in the Hidden State?

Karpathy's character-level RNN — hidden units learn to track:

- Open/close quotation marks
- Line position (newline counting)
- Whether inside a comment vs. code
- Indentation level

Heatmap of hidden unit activations:



Key Point

These features emerge **without explicit supervision** — the model discovers useful representations through training.

Limitations of the Hidden State

The Bottleneck

Fixed-size vector (e.g., 256 dims) must encode **everything** seen so far.

- Older information gradually gets **overwritten** by newer information
- The network has finite memory capacity
- Long sequences = early tokens get “forgotten”

Analogy

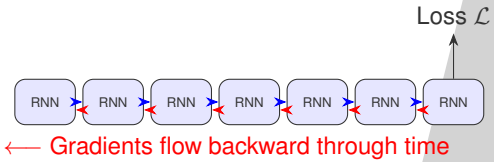
“Like a sticky note you keep erasing and rewriting. Eventually, early information is lost.”

This leads directly to the vanishing gradient problem. . .

BPTT: The Big Idea

Three Steps

1. **Unroll** the RNN across all T timesteps
2. The unrolled RNN looks like a **very deep feedforward network**
3. Apply **standard backpropagation** on this unrolled graph



"A 100-token sequence = backprop through a 100-layer network!"

The Chain Rule Through Time

The gradient of the loss w.r.t. W_{hh} involves:

$$\frac{\partial \mathcal{L}}{\partial W_{hh}} \propto \prod_{t=k}^{T} \frac{\partial h_t}{\partial h_{t-1}}$$

Each factor: $\frac{\partial h_t}{\partial h_{t-1}} = W_{hh}^T \cdot \text{diag}(\tanh'(\cdot))$

The Problem

- If largest singular value of $W_{hh} < 1$: product $\to 0$ (**vanishing**)
- If largest singular value of $W_{hh} > 1$: product $\to \infty$ (**exploding**)

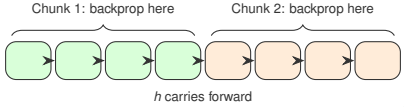
Repeated multiplication makes things vanish or explode.

Truncated BPTT

Problem

Full BPTT on 10,000 tokens = backprop through 10,000 layers. Impractical!

Solution: Only unroll for K steps (e.g., $K = 35$)

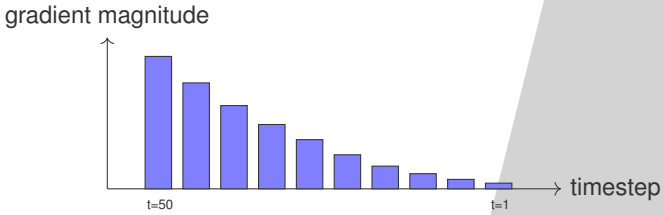


- Process sequence in chunks, carry h forward
- Only backpropagate within each chunk
- Trade-off: can't learn dependencies longer than K steps
- Common values: $K = 20$ to $K = 100$

The Vanishing Gradient Problem

Core idea: Multiply a number < 1 by itself many times $\rightarrow 0$

Steps	0.9^n
10	0.349
20	0.122
50	0.005
100	0.00003



The gradient from the loss barely reaches early timesteps!

Why This Kills Long-Range Learning

The Problem in Practice

The network **cannot learn** that an input from 50 steps ago matters for the current prediction.

Example

"The students who came from France _____ French."

- Answer depends on "students" (subject) and "France" (context)
- Both are far from the blank
- Gradient from the blank **can't reach back** to "students" — vanishes!

In practice: Vanilla RNNs fail on dependencies beyond ∼20 tokens.

Exploding Gradients and Gradient Clipping

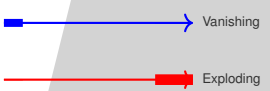
Exploding: If singular values > 1, gradients grow exponentially.

Symptoms: Loss $\rightarrow$ NaN, parameters $\rightarrow \infty$

Fix: Gradient Clipping

- If $\|\nabla\| >$ threshold, scale it down
- Simple, effective, widely used

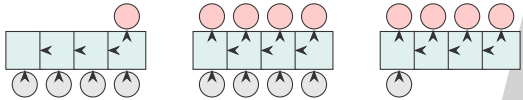
```
torch.nn.utils.clip_grad_norm_(
  model.parameters(), max_norm=5.0)
```



Vanishing = Unsolved

Clipping fixes exploding.
Vanishing needs **new architecture**.
$\rightarrow$ LSTMs (next week!)

The Four RNN Patterns



Many-to-One **Many-to-Many** **One-to-Many**

Pattern	Input/Output	Example Tasks
Many-to-One	Sequence $\rightarrow$ Label	Sentiment, classification
Many-to-Many	Sequence $\rightarrow$ Sequence	POS tagging, NER
One-to-Many	Single $\rightarrow$ Sequence	Image captioning, music gen.

Final Quiz

1. Why do gradients vanish in RNNs?
   - ✓ Repeated multiplication by W_{hh} with singular values < 1 causes exponential decay
2. Name a many-to-one task.
   - ✓ Sentiment analysis, document classification, spam detection
3. What is truncated BPTT?
   - ✓ Only backpropagate through K timesteps instead of the full sequence — trades long-range learning for computational feasibility

Recap & What's Next

Today's Key Takeaways

- Sequences need architectures that handle **variable length + order**
- RNNs **share weights** across time, maintain a hidden state as memory
- Trained with **BPTT** (unroll + backprop)
- **Vanishing gradients** limit learning beyond $\sim$20 tokens

Next Session: LSTMs and GRUs

Architectures with **gates** that create "gradient highways" through time — solving the vanishing gradient problem.

"If RNNs are the idea, LSTMs are the engineering that made it work."

Thank You!

Questions?

Priyansh Saxena

All lecture materials available at:

www.priyanshsaxena.com/sst-deep-learning-nn