Recurrent Neural Networks

Nipun Batra and teaching staff
IIT Gandhinagar

August 3, 2025

Table of Contents

Why Sequential Data Matters

Example: Sequential Data Examples

- **Text**: "The quick brown fox jumps..."

Important: Challenge

Traditional feedforward networks treat inputs independently - they can't capture **temporal dependencies**.

Why Sequential Data Matters

Example: Sequential Data Examples

- **Text**: "The quick brown fox jumps..."
- **Speech**: Audio waveforms over time

Important: Challenge

Traditional feedforward networks treat inputs independently - they can't capture **temporal dependencies**.

Why Sequential Data Matters

Example: Sequential Data Examples

- **Text**: "The quick brown fox jumps..."
- **Speech**: Audio waveforms over time
- **Stock Prices**: Daily market values

Important: Challenge

Traditional feedforward networks treat inputs independently - they can't capture **temporal dependencies**.

Why Sequential Data Matters

Example: Sequential Data Examples

- **Text**: "The quick brown fox jumps..."
- **Speech**: Audio waveforms over time
- **Stock Prices**: Daily market values
- **Weather**: Temperature, humidity over days

Important: Challenge

Traditional feedforward networks treat inputs independently - they can't capture **temporal dependencies**.

Simple RNN Cell

Definition: RNN Equations

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \quad (1)$$
$$y_t = W_{hy}h_t + b_y \quad (2)$$

Key Points

- Same weights shared across all time steps

Simple RNN Cell

Definition: RNN Equations

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

$$y_t = W_{hy}h_t + b_y \tag{2}$$

Key Points

- Same weights shared across all time steps
- Hidden state acts as "memory"

Simple RNN Cell

Definition: RNN Equations

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

$$y_t = W_{hy}h_t + b_y \quad (2)$$

Key Points

- Same weights shared across all time steps
- Hidden state acts as "memory"
- Can process variable length sequences

Pop Quiz #1

Quick Quiz 1

What happens to gradients in simple RNNs during backpropagation?

A) They remain constant

Pop Quiz #1

Quick Quiz 1

What happens to gradients in simple RNNs during backpropagation?

A) They remain constant

B) They can explode or vanish

Pop Quiz #1

Quick Quiz 1

What happens to gradients in simple RNNs during backpropagation?

A) They remain constant
B) They can explode or vanish
C) They always improve

Pop Quiz #1

Quick Quiz 1

What happens to gradients in simple RNNs during backpropagation?

A) They remain constant
B) They can explode or vanish
C) They always improve
D) They disappear completely

Pop Quiz #1 - Answer

Answer: B) They can explode or vanish

Important: The Gradient Problem

- Gradients multiply by W_{hh} at each time step

Pop Quiz #1 - Answer

Answer: B) They can explode or vanish

Important: The Gradient Problem

- Gradients multiply by W_{hh} at each time step
- If $||W_{hh}|| > 1$: Exploding gradients

Pop Quiz #1 - Answer

Answer: B) They can explode or vanish

Important: The Gradient Problem

- Gradients multiply by W_{hh} at each time step
- If $||W_{hh}|| > 1$: Exploding gradients
- If $||W_{hh}|| < 1$: Vanishing gradients

Sentiment Analysis (Many-to-One)

Example: Sequence Classification

- Input: "This movie is great!"

Key Points

Applications: Document classification, spam detection, review analysis

Sentiment Analysis (Many-to-One)

Example: Sequence Classification

- Input: "This movie is great!"
- Process each word sequentially

Key Points

Applications: Document classification, spam detection, review analysis

Sentiment Analysis (Many-to-One)

Example: Sequence Classification

- Input: "This movie is great!"
- Process each word sequentially
- Output: Positive/Negative sentiment

Key Points

Applications: Document classification, spam detection, review analysis

Machine Translation (Many-to-Many)

Example: Sequence-to-Sequence

- **Encoder**: French → "Je suis étudiant"

Machine Translation (Many-to-Many)

Example: Sequence-to-Sequence

- **Encoder**: French → "Je suis étudiant"
- **Context**: Hidden representation

Machine Translation (Many-to-Many)

Example: Sequence-to-Sequence

- **Encoder**: French → "Je suis étudiant"
- **Context**: Hidden representation
- **Decoder**: English → "I am student"

LSTM: Long Short-Term Memory

Definition: LSTM Key Idea

Use **gates** to control information flow:

- **Forget gate**: What to remove from memory

Theorem: Advantage

LSTM gates solve the vanishing gradient problem by allowing gradients to flow unchanged through time.

LSTM: Long Short-Term Memory

Definition: LSTM Key Idea

Use **gates** to control information flow:

- **Forget gate**: What to remove from memory
- **Input gate**: What new information to store

Theorem: Advantage

LSTM gates solve the vanishing gradient problem by allowing gradients to flow unchanged through time.

LSTM: Long Short-Term Memory

Definition: LSTM Key Idea

Use **gates** to control information flow:

- **Forget gate**: What to remove from memory
- **Input gate**: What new information to store
- **Output gate**: What parts of memory to output

Theorem: Advantage

LSTM gates solve the vanishing gradient problem by allowing gradients to flow unchanged through time.

GRU: Gated Recurrent Unit

Key Points

GRU vs LSTM:

- Simpler: Only 2 gates instead of 3

GRU: Gated Recurrent Unit

Key Points

GRU vs LSTM:

- Simpler: Only 2 gates instead of 3
- Faster training and inference

GRU: Gated Recurrent Unit

Key Points

GRU vs LSTM:

- Simpler: Only 2 gates instead of 3
- Faster training and inference
- Often performs similarly to LSTM

GRU: Gated Recurrent Unit

Key Points

GRU vs LSTM:

- Simpler: Only 2 gates instead of 3
- Faster training and inference
- Often performs similarly to LSTM
- Good starting point for many applications

From RNNs to Transformers

Theorem: Why Transformers Won

- **Parallelizable**: No sequential dependency

From RNNs to Transformers

Theorem: Why Transformers Won

- **Parallelizable**: No sequential dependency
- **Long-range dependencies**: Attention mechanism

From RNNs to Transformers

Theorem: Why Transformers Won

- **Parallelizable**: No sequential dependency
- **Long-range dependencies**: Attention mechanism
- **Scalable**: Works well with large datasets

From RNNs to Transformers

Theorem: Why Transformers Won

- **Parallelizable**: No sequential dependency
- **Long-range dependencies**: Attention mechanism
- **Scalable**: Works well with large datasets
- **Transfer learning**: Pre-trained models (GPT, BERT)

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage

Example: Modern Applications

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data

Example: Modern Applications

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data
- **Small datasets**: Less prone to overfitting

Example: Modern Applications

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data
- **Small datasets**: Less prone to overfitting
- **Simple problems**: Often sufficient

Example: Modern Applications

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data
- **Small datasets**: Less prone to overfitting
- **Simple problems**: Often sufficient

Example: Modern Applications

- Real-time speech recognition

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data
- **Small datasets**: Less prone to overfitting
- **Simple problems**: Often sufficient

Example: Modern Applications

- Real-time speech recognition
- IoT sensor data processing

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data
- **Small datasets**: Less prone to overfitting
- **Simple problems**: Often sufficient

Example: Modern Applications

- Real-time speech recognition
- IoT sensor data processing
- Mobile applications

When to Still Use RNNs

Definition: RNN Strengths

- **Memory efficiency**: Constant memory usage
- **Online processing**: Can process streaming data
- **Small datasets**: Less prone to overfitting
- **Simple problems**: Often sufficient

Example: Modern Applications

- Real-time speech recognition
- IoT sensor data processing
- Mobile applications
- Control systems

Key Takeaways

Key Points

What we learned:

1. RNNs process sequential data with memory

Theorem: The Big Picture

RNNs introduced **sequential processing with memory** to deep learning, paving the way for modern language models.

Key Takeaways

Key Points

What we learned:

1. RNNs process sequential data with memory
2. Simple RNNs suffer from gradient problems

Theorem: The Big Picture

RNNs introduced **sequential processing with memory** to deep learning, paving the way for modern language models.

Key Takeaways

Key Points

What we learned:

1. RNNs process sequential data with memory
2. Simple RNNs suffer from gradient problems
3. LSTM and GRU solve long-term dependencies

Theorem: The Big Picture

RNNs introduced **sequential processing with memory** to deep learning, paving the way for modern language models.

Key Takeaways

Key Points

What we learned:

1. RNNs process sequential data with memory
2. Simple RNNs suffer from gradient problems
3. LSTM and GRU solve long-term dependencies
4. Training uses Backpropagation Through Time

Theorem: The Big Picture

RNNs introduced **sequential processing with memory** to deep learning, paving the way for modern language models.

Key Takeaways

Key Points

What we learned:

1. RNNs process sequential data with memory
2. Simple RNNs suffer from gradient problems
3. LSTM and GRU solve long-term dependencies
4. Training uses Backpropagation Through Time
5. Transformers have largely replaced RNNs for NLP

Theorem: The Big Picture

RNNs introduced **sequential processing with memory** to deep learning, paving the way for modern language models.