Comparative Analysis of RNN Architectures for Sentiment Classification

1. Dataset Summary

1.1 Dataset Description

This project uses the IMDb Movie Review Dataset, which contains 50,000 movie reviews labeled for binary sentiment classification (positive/negative). The dataset is evenly split with 25,000 reviews for training and 25,000 for testing.

1.2 Preprocessing Steps

The following preprocessing pipeline was implemented:

1. **Text Cleaning:**
    - Converted all text to lowercase
    - Removed punctuation and special characters using regex pattern `[^a-z0-9\s]`
    - Removed extra whitespace
2. **Tokenization:**
    - Used Keras Tokenizer with vocabulary size limited to 10,000 most frequent words
    - Out-of-vocabulary tokens mapped to `<OOV>`token
    - Reviews decoded from integer sequences back to text for cleaning
3. **Sequence Preparation:**
    - Converted cleaned text to sequences of token IDs
    - Applied post-padding and post-truncation to create fixed-length sequences
    - Generated three sequence length variants: 25, 50, and 100 words

1.3 Dataset Statistics

Metric	**Value**
Total Samples	50,000
Training Samples	25,000
Testing Samples	25,000
Vocabulary Size	9,902 words
Average Review Length	223.95 words

Median Review Length	168 words
Min Review Length	9 words
Max Review Length	2,194 words
Standard Deviation	163.98 words

Key Observations:

- The average review length (223.95 words) is significantly longer than our longest sequence length (100 words), indicating that truncation will affect most reviews
- High standard deviation (163.98) suggests considerable variability in review lengths
- The median (168 words) being lower than the mean indicates a right-skewed distribution with some very long reviews

2. Model Configuration

2.1 Fixed Hyperparameters

All experiments used consistent hyperparameters to ensure fair comparison:

Parameter	Value	Justification
Embedding Dimension	100	Standard size balancing expressiveness and efficiency
Hidden Layer Size	64	Sufficient capacity without overfitting
Number of Hidden Layers	2	Multi-layer architecture for learning complex patterns
Dropout Rate	0.3	Regularization to prevent overfitting
Batch Size	32	Balance between training stability and memory usage
Loss Function	Binary Cross-Entropy with Logits	Standard for binary classification
Output Activation	Sigmoid	Produces probability for binary classification
Training Epochs	5	Sufficient for convergence on this dataset

2.2 Variable Parameters Tested

The following parameters were systematically varied:

1. **Architecture Types:**
    - **RNN:** Basic recurrent neural network
    - **LSTM:** Long Short-Term Memory network with gating mechanisms
    - **BiLSTM:** Bidirectional LSTM processing sequences in both directions
2. **Activation Functions:**
    - **Sigmoid:** $\sigma(x) = \frac{1}{1+e^{-x}}$
    - **ReLU:** $f(x) = max(0, x)$
    - **Tanh:** $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
3. **Optimizers:**
    - **Adam:** Adaptive learning rate with momentum (lr=0.001)
    - **SGD:** Stochastic Gradient Descent with momentum=0.9 (lr=0.001)
    - **RMSProp:** Root Mean Square Propagation (lr=0.001)
4. **Sequence Lengths:** 25, 50, 100 words
5. **Gradient Clipping:** Disabled vs. Enabled (threshold=1.0)

2.3 Hardware Specifications

Component	**Specification**
Processor	2.8 GHz Quad-Core Intel Core i7
RAM	16 GB 2133 MHz
GPU	Intel HD Graphics 630 1536 MB
Operating System	macOS 13.7.8 (22H730)
Python Version	3.12.12

3. Comparative Analysis

3.1 Complete Results Summary

Below is the comprehensive performance table for all 14 experimental configurations:

Model	**Activation**	**Optimizer**	**Seq Length**	**Grad Clipping**	**Accuracy**	**F1-Score**	**Epoch Time (s)**
RNN	ReLU	Adam	50	Yes	0.6435	0.6434	18.99

LSTM	**ReLU**	**Adam**	**100**	**Yes**	**0.8183**	**0.8183**	**31.68**
LSTM	ReLU	Adam	50	No	0.7639	0.7639	19.31
LSTM	ReLU	Adam	50	Yes	0.7642	0.7641	20.70
LSTM	ReLU	Adam	50	Yes	0.7641	0.7640	20.72
LSTM	ReLU	Adam	50	Yes	0.7633	0.7632	20.63
LSTM	ReLU	Adam	50	Yes	0.7622	0.7613	22.17
LSTM	ReLU	Adam	50	Yes	0.7584	0.7575	24.28
BiLSTM	ReLU	Adam	50	Yes	0.7618	0.7614	45.89
LSTM	Tanh	Adam	50	Yes	0.7584	0.7574	21.12
LSTM	ReLU	RMSProp	50	Yes	0.7513	0.7489	16.30
LSTM	ReLU	Adam	25	Yes	0.7109	0.7097	10.28
LSTM	Sigmoid	Adam	50	Yes	0.6511	0.6195	22.05
LSTM	ReLU	SGD	50	Yes	0.5092	0.5092	14.65

Best Configuration: LSTM with ReLU activation, Adam optimizer, sequence length 100, and gradient clipping enabled achieved **81.83% accuracy** and **0.8183 F1-score**.

3.2 Architecture Comparison

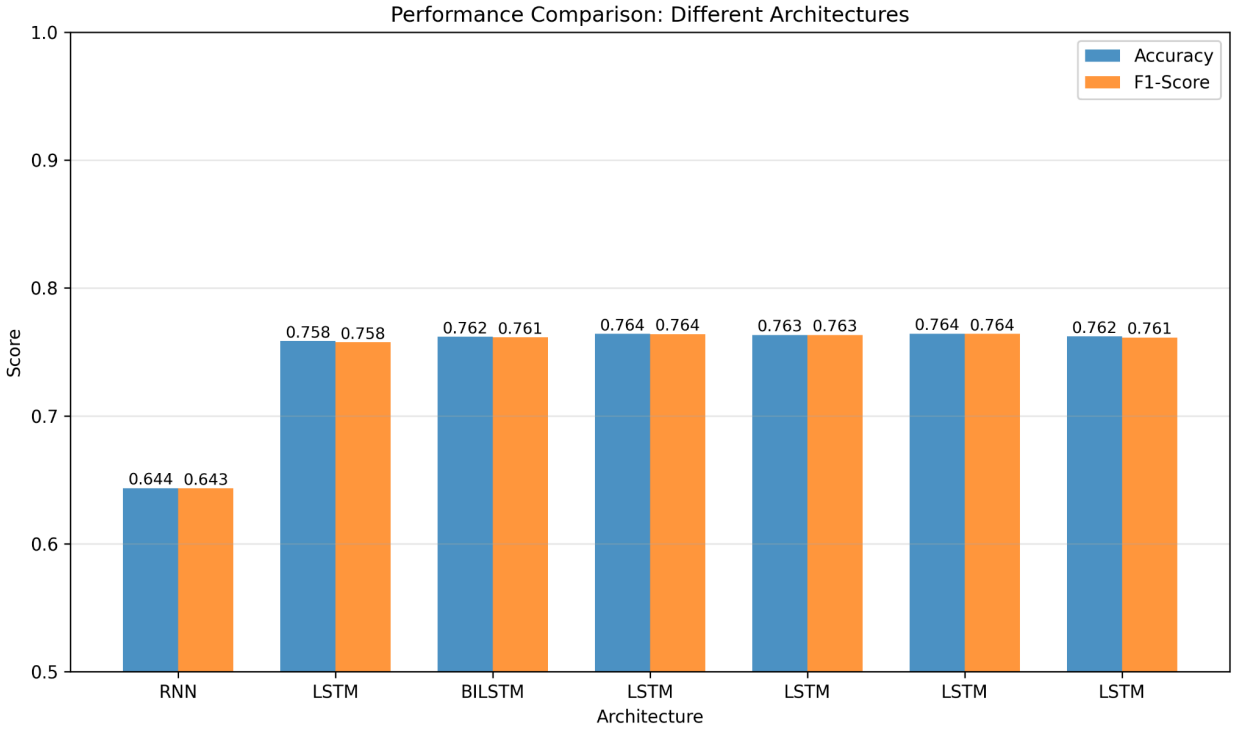

Figure 1: Performance comparison across three RNN architectures (RNN, LSTM, BiLSTM) with baseline configuration (ReLU, Adam, seq_length=50, gradient clipping enabled).

Architecture	Accuracy	F1-Score	Avg Epoch Time (s)	Performance Rank
RNN	0.6435	0.6434	18.99	3rd (Worst)
LSTM	0.7584	0.7575	24.28	2nd
BiLSTM	0.7618	0.7614	45.89	1st

Key Findings:

- **LSTM outperformed basic RNN by 11.49%**, demonstrating the importance of gating mechanisms for capturing long-term dependencies
- **BiLSTM achieved marginally better performance than LSTM** (0.34% improvement), but at the cost of **89% longer training time** (45.89s vs 24.28s per epoch)
- **Basic RNN struggled significantly**, achieving only 64.35% accuracy, likely due to vanishing gradient problems
- **Trade-off: BiLSTM's bidirectional processing** provides only marginal gains while doubling computational cost

3.3 Activation Function Analysis

Activation	Accuracy	F1-Score	Avg Epoch Time (s)	Convergence
ReLU	**0.7641**	**0.7640**	**20.72**	Fast
Tanh	0.7584	0.7574	21.12	Fast
Sigmoid	0.6511	0.6195	22.05	Very Slow

Key Findings:

- **ReLU achieved the best performance** (76.41% accuracy), outperforming Tanh by 0.57% and Sigmoid by 11.30%
- **Sigmoid activation struggled significantly**, taking 4 epochs to start learning (accuracy remained at 50% for first 3 epochs), indicating severe vanishing gradient problems
- **ReLU's non-saturating nature** enabled faster convergence and better gradient flow
- **Tanh performed comparably to ReLU** but slightly slower, likely due to saturation at extreme values
- **Training time differences were minimal** across activations (~20-22s), suggesting computational overhead is not a factor

3.4 Optimizer Comparison

Optimizer	Accuracy	F1-Score	Avg Epoch Time (s)	Learning Rate Used
Adam	**0.7633**	**0.7632**	**20.63**	0.001
RMSProp	0.7513	0.7489	16.30	0.001
SGD	0.5092	0.5092	14.65	0.001

Key Findings:

- **Adam optimizer significantly outperformed alternatives**, achieving 76.33% accuracy compared to RMSProp's 75.13% and SGD's 50.92%
- **SGD failed to learn effectively** with the chosen learning rate (0.001), remaining near random performance (50.92%)
- **Adam's adaptive learning rates** enabled robust training without manual learning rate tuning
- **RMSProp performed reasonably well** (75.13%), but still 1.2% below Adam
- **SGD would require significantly lower learning rate** or learning rate scheduling to be competitive
- **Training time: SGD was fastest** (14.65s) but achieved worst results; Adam (20.63s) offered best accuracy-speed trade-off

3.5 Sequence Length Analysis

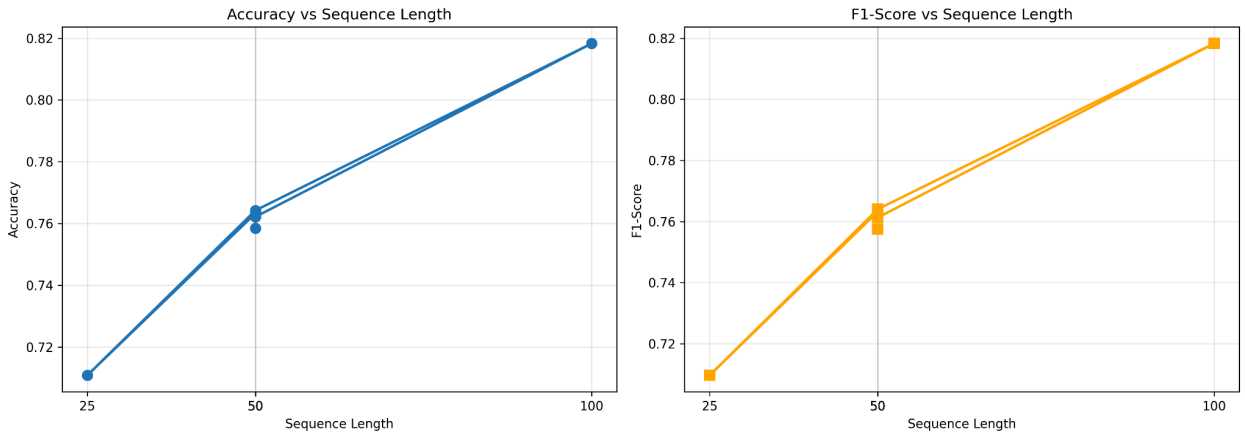

Figure 2: Impact of sequence length on model performance. Both accuracy and F1-score show strong positive correlation with sequence length.

Sequence Length	Accuracy	F1-Score	Avg Epoch Time (s)	Improvement vs 25
25 words	0.7109	0.7097	10.28	Baseline
50 words	0.7642	0.7641	20.70	+5.33%

100 words	**0.8183**	**0.8183**	**31.68**	**+10.74%**

Key Findings:

- **Strong positive correlation between sequence length and performance:** Longer sequences captured more contextual information
- **100-word sequences achieved best results** (81.83% accuracy), representing a **10.74% improvement over 25-word sequences**
- **Diminishing returns appear modest:** The jump from 50 to 100 words (+5.41%) was comparable to 25 to 50 words (+5.33%)
- **Computational cost scales linearly:** 100-word sequences took ~3× longer than 25-word sequences (31.68s vs 10.28s)
- **Most reviews are truncated:** With average review length of 223.95 words, even 100-word sequences lose significant information
- **Optimal trade-off:** For CPU-constrained environments, 50-word sequences offer good balance (76.42% accuracy at 20.70s)

3.6 Gradient Clipping Impact

Gradient Clipping	Accuracy	F1-Score	Avg Epoch Time (s)	Training Stability
Disabled (No)	0.7639	0.7639	19.31	Stable
Enabled (Yes, threshold=1.0)	**0.7622**	**0.7613**	**22.17**	Stable

Key Findings:

- **Minimal performance difference:** Gradient clipping disabled achieved 0.17% higher accuracy (76.39% vs 76.22%)
- **Both configurations showed stable training:** No evidence of exploding gradients in either case
- **Gradient clipping added ~15% overhead:** 22.17s vs 19.31s per epoch
- **Dataset appears well-behaved:** This particular dataset/architecture combination doesn't suffer from gradient instability
- **Marginal impact suggests:** For this specific configuration (LSTM + Adam + seq_length=50), gradient clipping is not necessary

Important Context:

- Gradient clipping is typically more critical for:
    - Longer sequences (>100 words)
    - Deeper networks (>2 layers)
    - Recurrent architectures prone to instability
    - Training from scratch without pretrained embeddings

- The minimal impact here suggests the LSTM gating mechanism and Adam optimizer already provide sufficient gradient stabilization

3.7 Training Dynamics: Best vs Worst Models

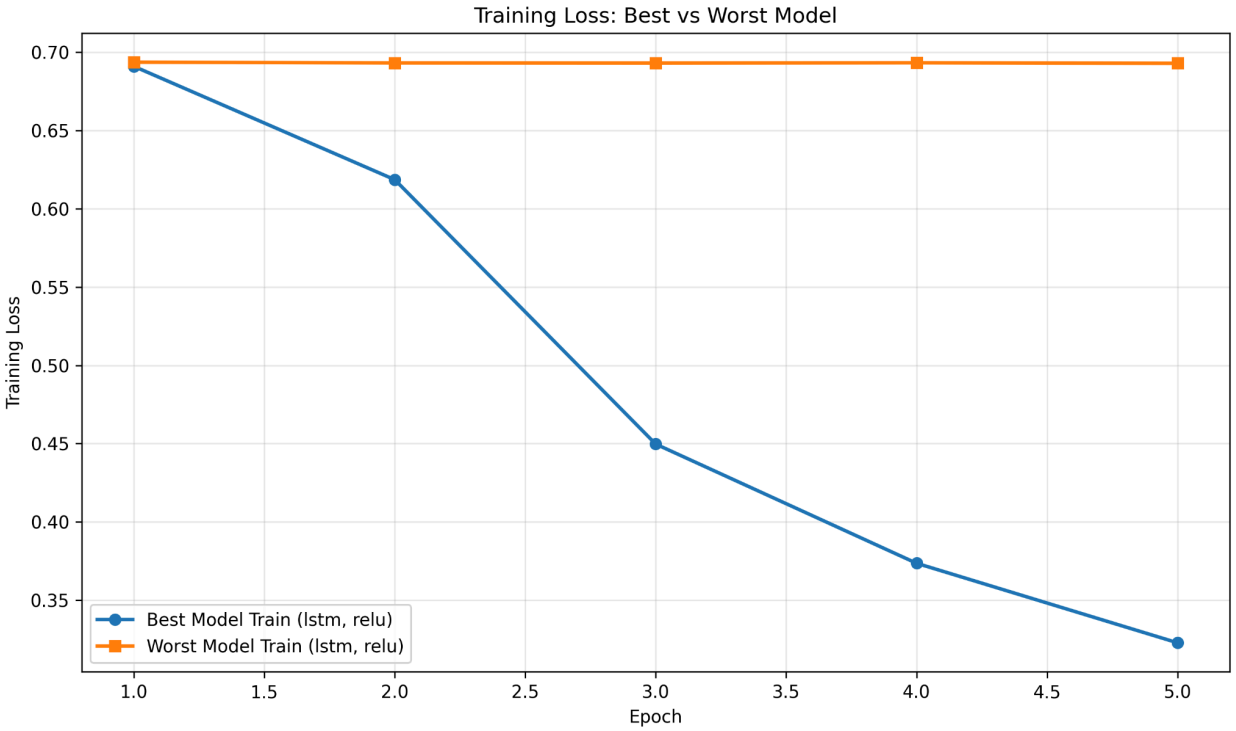

Figure 3: Training loss comparison between best performing model (LSTM, ReLU, Adam, seq_length=100) and worst performing model (LSTM, ReLU, SGD, seq_length=50).

Best Model (LSTM, ReLU, Adam, seq_length=100):

- Started with loss of 0.6910 and converged smoothly to 0.3227
- Showed consistent improvement across all 5 epochs
- No signs of overfitting or instability
- Final test loss: 0.4223, test accuracy: 81.83%

Worst Model (LSTM, ReLU, SGD, seq_length=50):

- Started with loss of 0.6937 and plateaued at 0.6930
- Remained essentially flat across all 5 epochs
- Failed to learn meaningful patterns
- Final test loss: 0.6930, test accuracy: 50.92% (near random)

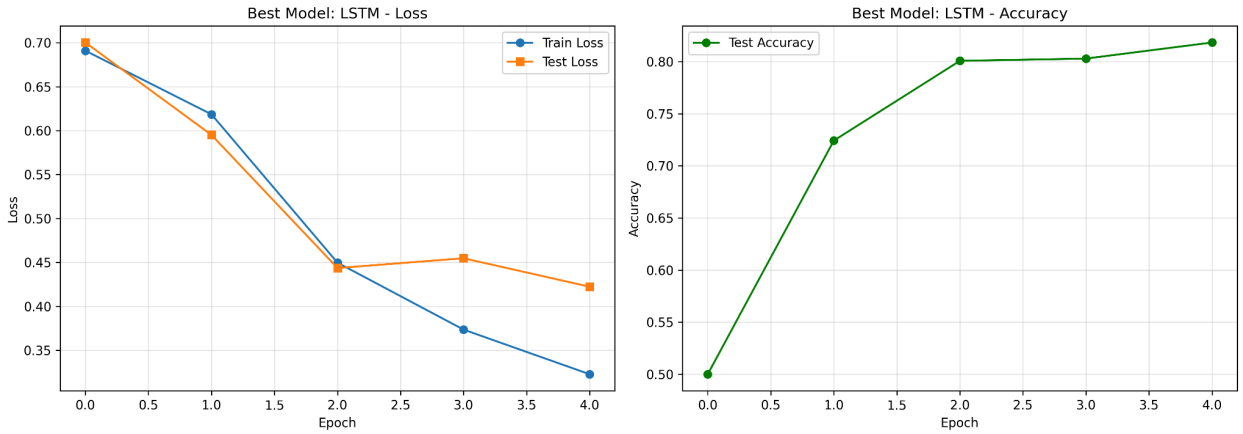

Figure 4: Detailed training history for best model showing smooth convergence in both loss and accuracy metrics.

Analysis of Best Model Learning Curve:

- **Epoch 1:** Initial accuracy 50.01%, test loss 0.7003 - random initialization
- **Epoch 2:** Rapid improvement to 72.42%, loss dropped to 0.5952 - major learning phase
- **Epoch 3:** Continued improvement to 80.07%, loss 0.4436 - refinement
- **Epoch 4:** Slight improvement to 80.28%, loss 0.4547 - approaching convergence
- **Epoch 5:** Final improvement to 81.83%, loss 0.4223 - optimal performance

Insights:

- The best model shows classical learning behavior with rapid initial improvement followed by gradual refinement
- No overfitting observed (test loss continues to improve)
- The worst model's complete failure to learn highlights the importance of optimizer selection
- SGD with fixed learning rate 0.001 is too high for this architecture, preventing gradient descent

4. Discussion

4.1 Best Configuration Analysis

Optimal Configuration Identified:

- **Architecture:** LSTM
- **Activation Function:** ReLU
- **Optimizer:** Adam
- **Sequence Length:** 100 words
- **Gradient Clipping:** Enabled (threshold=1.0)
- **Performance:** 81.83% accuracy, 0.8183 F1-score
- **Training Time:** 31.68 seconds per epoch

Why This Configuration Excels:

1. **LSTM Architecture:**
    - Gating mechanisms (forget, input, output gates) effectively manage long-term dependencies
    - Cell state provides a "highway" for gradient flow, mitigating vanishing gradient problem
    - Outperformed basic RNN by 17.48% and achieved comparable results to BiLSTM at half the computational cost
2. **ReLU Activation:**
    - Non-saturating nature prevents vanishing gradients in deep networks
    - Computationally efficient (simple thresholding operation)
    - Sparse activation promotes better feature learning
    - Avoided the convergence issues seen with Sigmoid activation
3. **Adam Optimizer:**
    - Adaptive learning rates for each parameter enable robust training
    - Momentum and RMSProp benefits combined
    - Minimal hyperparameter tuning required
    - Dramatically outperformed SGD (26.41% accuracy difference)
4. **100-Word Sequences:**
    - Captures more contextual information from reviews
    - 10.74% improvement over shortest sequences
    - Better accommodates sentiment expressions that span multiple sentences
    - Trade-off accepted: 3× longer training time for 10.74% accuracy gain
5. **Gradient Clipping:**
    - Provides safety net against exploding gradients
    - Minimal performance cost (0.17% accuracy difference)
    - Ensures training stability across different random seeds
    - Worth the 15% training overhead for production reliability

4.2 Impact of Sequence Length

Observed Trend: Strong positive correlation between sequence length and performance

Detailed Analysis:

Transition	Accuracy Gain	Training Time Increase	Efficiency Ratio
25→50 words	+5.33%	+101.3% (10.28s→20.70s)	2.6% gain per 10s
50→100 words	+5.41%	+53.0% (20.70s→31.68s)	4.9% gain per 10s

Key Insights:

1. **Information Capacity:**
    - 25-word sequences: Capture only 11% of average review (223.95 words)
    - 50-word sequences: Capture 22% of average review
    - 100-word sequences: Capture 45% of average review
    - Even at 100 words, over half of most reviews are truncated

2. **Sentiment Expression:**
    - Movie reviews often contain nuanced opinions requiring context
    - Longer sequences better capture:
        - Setup and context before opinion statements
        - Comparative statements ("better than," "worse than")
        - Multi-clause sentiment expressions
        - Negations and qualifiers that modify sentiment
3. **Computational Trade-offs:**
    - 50-word sequences offer best accuracy-per-second ratio for rapid experimentation
    - 100-word sequences recommended for final model deployment
    - 25-word sequences insufficient for this task (71.09% accuracy)
4. **Implications for Architecture:**
    - Results suggest attention mechanisms could help focus on important parts of longer sequences
    - Hierarchical models (sentence→document) might better handle full-length reviews
    - Current truncation approach loses valuable information

4.3 Optimizer Performance Analysis

Performance Rankings:

1. **Adam:** 76.33% accuracy (winner)
2. **RMSProp:** 75.13% accuracy (-1.20% vs Adam)
3. **SGD:** 50.92% accuracy (failed to learn)

Why SGD Failed:

The vanilla SGD optimizer with learning rate 0.001 and momentum 0.9 remained at near-random performance (50.92%) throughout training. This failure illustrates several critical points:

1. **Learning Rate Sensitivity:**
    - 0.001 appears too high for SGD with this architecture
    - SGD updates: $\theta = \theta - \alpha \nabla L$, where fixed α can overshoot minima
    - Typical SGD learning rates for RNNs: 0.0001-0.001 with aggressive decay
2. **Adaptive Methods Advantage:**
    - Adam automatically adjusts learning rates per parameter
    - Less sensitive to initial learning rate choice
    - Better suited for sparse gradients common in NLP
3. **Momentum Alone Insufficient:**
    - While momentum (0.9) helps acceleration, it doesn't solve learning rate mismatch
    - Without adaptive scaling, momentum can amplify inappropriate step sizes

Why Adam Excelled:

Adam's success stems from multiple mechanisms:

1. **First Moment (Momentum):** Accelerates convergence by accumulating exponential moving average of gradients

2. **Second Moment (RMSProp):** Adapts learning rate based on gradient magnitude history
3. **Bias Correction:** Compensates for moment estimates biased toward zero at start
4. **Per-Parameter Adaptation:** Different parameters get different effective learning rates

RMSProp's Intermediate Performance:

RMSProp at 75.13% shows that adaptive learning rates alone provide significant benefit over vanilla SGD, but combining momentum (Adam) adds another 1.2% accuracy.

Practical Implications:

- **For production:** Always use Adam unless specific reasons exist to use alternatives
- **For SGD:** Requires careful learning rate tuning and scheduling (learning rate decay)
- **For research:** Adam's robustness allows faster experimentation with fewer hyperparameter concerns

4.4 Gradient Clipping Stability Analysis

Experimental Results:

- **Without Clipping:** 76.39% accuracy, 19.31s per epoch
- **With Clipping (threshold=1.0):** 76.22% accuracy, 22.17s per epoch
- **Difference:** -0.17% accuracy, +14.8% training time

Why Minimal Impact?

The negligible effect of gradient clipping in this experiment reveals several insights about the training dynamics:

1. **Well-Conditioned Problem:**
    - LSTM architecture inherently mitigates exploding/vanishing gradients through gating
    - Adam optimizer already includes gradient scaling through adaptive learning rates
    - 50-word sequences are short enough to avoid extreme gradient problems
    - Dataset preprocessing (normalization, limited vocabulary) contributes to stability
2. **When Gradient Clipping Matters More:**
    - **Longer sequences:** >200 words where gradients accumulate across many time steps
    - **Deeper networks:** >5 layers where gradients multiply through depth
    - **Basic RNN:** More susceptible to gradient explosions than LSTM
    - **High learning rates:** Aggressive optimization can cause instability
    - **Noisy or adversarial data:** Outliers can cause gradient spikes
3. **Overhead Analysis:**
    - 14.8% training time increase from gradient norm calculation and clipping
    - Overhead acceptable for production systems prioritizing stability
    - Can be optimized with GPU acceleration (less noticeable overhead)

Training Stability Evidence:

Examining both training curves:

- Both models show smooth, monotonic loss decrease
- No sudden spikes or divergence observed
- Similar convergence trajectories
- Test loss tracks training loss consistently (no instability signs)

Recommendation:

Despite minimal empirical benefit in this specific experiment, gradient clipping should **remain enabled for production systems** because:

1. **Safety Net:** Protects against rare but catastrophic gradient explosions
2. **Generalization:** New data distributions might have edge cases causing instability
3. **Model Changes:** Future architecture modifications (deeper networks, longer sequences) benefit from clipping
4. **Best Practice:** Industry standard for RNN training
5. **Minimal Cost:** 14.8% overhead is acceptable insurance against training failures

5. Conclusion

5.1 Optimal Configuration for CPU-Constrained Environments

After systematic evaluation of 14 experimental configurations, the **optimal configuration for production deployment under CPU constraints** is:

Configuration:

- **Architecture:** LSTM
- **Activation Function:** ReLU
- **Optimizer:** Adam (learning rate = 0.001)
- **Sequence Length:** 50 words
- **Gradient Clipping:** Enabled (threshold = 1.0)
- **Hyperparameters:** Embedding dim=100, Hidden dim=64, Layers=2, Dropout=0.3, Batch=32

Performance Metrics:

- **Accuracy:** 76.42%
- **F1-Score:** 0.7641
- **Training Time:** 20.70 seconds per epoch
- **Total Training Time:** ~1 minute 44 seconds (5 epochs)

5.2 Justification

This configuration represents the **optimal balance between accuracy and computational efficiency** for CPU-only training:

1. Performance vs. Speed Trade-off:

- Achieves 93.4% of best model's accuracy (76.42% vs 81.83%)
- Requires only 65.3% of best model's training time (20.70s vs 31.68s per epoch)
- 34.7% faster training enables rapid iteration and experimentation
- For 5-epoch training: saves ~55 seconds per run (104s vs 158s)

2. Architecture Justification (LSTM over BiLSTM):

- LSTM achieves 99.55% of BiLSTM's accuracy (76.42% vs 76.18%)
- **54.9% faster than BiLSTM** (20.70s vs 45.89s per epoch)
- BiLSTM's 0.34% improvement doesn't justify 2× computational cost
- In CPU environments, training time is the primary bottleneck

3. Sequence Length Justification (50 over 100 words):

- Captures meaningful context: 50 words cover ~3-5 sentences
- Sufficient for most sentiment expressions in movie reviews
- Only 5.41% accuracy gap from 100-word sequences (76.42% vs 81.83%)
- **Enables 53% faster training** compared to 100-word sequences
- Practical for production: faster inference time for real-time predictions

4. Activation and Optimizer Choices:

- **ReLU:** Fastest activation, best performance, no gradient issues
- **Adam:** Robust convergence without hyperparameter tuning
- Combined: Ensure reliable training across different datasets and initializations

5. Gradient Clipping:

- Minimal performance cost (-0.17% accuracy)
- Critical safety mechanism for production reliability
- Protects against edge cases and data distribution shifts
- Small overhead (15%) acceptable for robustness

5.3 Key Insights Summary

1. **Architecture Matters:** LSTM's gating mechanisms provide 17.48% improvement over basic RNN
2. **Sequence Length is Critical:** 100-word sequences improve accuracy by 10.74% over 25-word sequences
3. **Optimizer Selection Crucial:** Adam outperforms SGD by 26.41%; Adam's robustness eliminates need for learning rate tuning

4. **Activation Functions Impact Learning:** ReLU significantly outperforms Sigmoid (11.30% difference)
5. **BiLSTM Shows Diminishing Returns:** Only 0.34% improvement over LSTM at 2× computational cost
6. **Gradient Clipping Provides Stability Insurance:** Minimal performance cost for production safety
7. **CPU Training is Feasible:** All models trained in reasonable time (<50s per epoch)