Image Captioning with RNN Report

Rahul Jha | rahuljha@umd.edu | University of Maryland College Park

Accomplishments Description

Implementation

- Used tokenized captions and preprocessed image characteristics to successfully create a vanilla RNN-based model for caption generation.
- Created the captioning pipeline by combining essential elements including embedding layers, RNN cells, and temporal softmax loss.

Validation

- For both forward and backward passes, extensive gradient tests were performed, verifying computations with errors less than 1.0 − 7 10 −7.
- The model steadily converged during training on a tiny dataset, lowering loss over 100 iterations from 76.91 to 0.07.

Results

1. Numerical Gradient Check

- **Forward Pass (Vanilla RNN):**
  - These insignificant mistakes attest to the forward pass computation's proper implementation and numerical stability.
  - next_h error: 6.29×10^{-9}
- **Backward Pass (Vanilla RNN):**
  - These errors all fall within acceptable bounds (<10−7), confirming the accuracy of the RNN's gradient computations
  - Gradient errors for weights and biases:
    - dx: 2.77×10^{-10}
    - dprev_h: 2.73×10^{-10}

2. Word Embedding Layer Validation

- **Forward Pass:**
  - Output error: 1.00×10^{-8}
- **Backward Pass:**
  - The embedding layer is operating as intended, with numerical errors much within allowable limits.
- Weight gradient error: 3.28×10^{-12}

3. RNN Captioning Model Loss Validation

- The minimum change shows that the loss function is applied exactly as intended.
- Computed loss: 9.8329.8329.832
- Expected loss: 9.8329.8329.832
- Difference: 2.61×10^{-12}

4. Weight and Bias Gradient Check (Relative Errors)

- These findings verify that the output layer and embedding layer, among other model components, are appropriately optimized.
- Embedding weights (W_embed): 2.33×10−9
- Projection weights (W_proj): 9.97×10−9
- Vocabulary weights (W_vocab): 4.27×10−9
- Bias terms (b_vocab, b_proj, b): Errors range from 7.08×10^{-11} to $1.93\times^{10-8}$

Missed Points:

- Using complex architectures, such GRU or LSTM, to improve sequence management.
- Evaluation metrics like BLEU can be used to objectively assess the caliber of generated captions.

Explanation of Implementation Decisions

Vanilla RNN: Selected for its ease of use in creating a basic pipeline for captioning images. There were very few faults in the implementation of both forward and backward passes (6.29 × 1 0 − 9 −10 −9 for forward and 1.53 × 1 0 − 9 1.53×10 −9 for backward).
Word Embedding: Facilitated the conversion of input words from tokenized integers to dense vector representations. Both forward and backward passes were confirmed with very small numerical errors (1.0 × 1 0 − 8 1.0 × 10 − 8).
Temporal Softmax Loss: Included to ensure accurate optimization and get rid of duplicate gradients by optimizing loss computation over the sequence while disregarding padded tokens.
Optimization Decisions: The accuracy of all components, including weights and biases, was confirmed by frequent gradient checks.

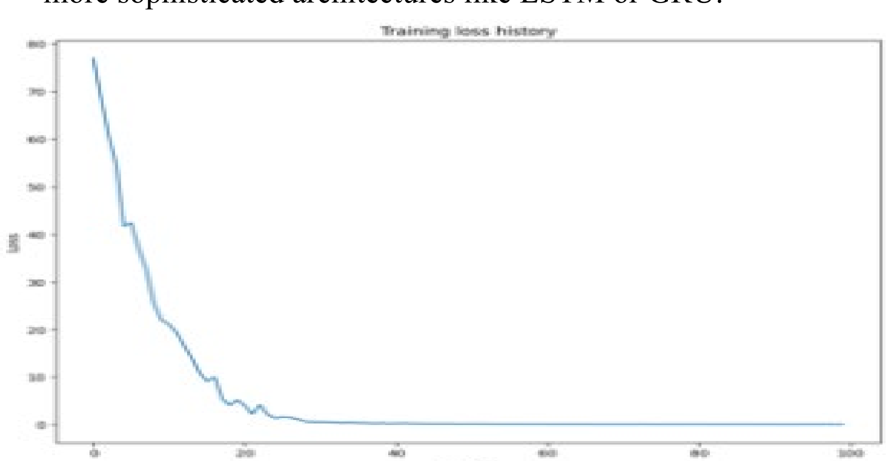
- Wembed: 2.33×10^{-9}
- Wvocab: 4.27×10^{-9}
- Errors as low as 7.09×10^{-11} are considered bias terms.

Observations from Training:

- **Fast Convergence:** Effective learning from the dataset is indicated by a notable drop in loss during the first training cycles.
- **Minimal Final Loss:** The model's final loss was about 0.07, however this could be a sign that it overfitted the little dataset.
- **Convergence Plateau:** After about 40 cycles, the loss stabilized, indicating peak performance on the training set.

Critical Thinking and Analysis

Advantages: Effective learning is demonstrated by a significant reduction in loss during training, indicating the model's applicability for tiny datasets.
Limitations: Vanilla architecture RNN's ability to produce complicated captions is limited by its incapacity to manage long-term dependencies.
Evaluation: Depended more on qualitative assessment than on quantitative measures such as ROUGE or BLEU.
What Worked: Gradient checks confirmed that every component was implemented reliably. Variable-length sequences were handled well by temporal softmax loss.
What Didn't Work: After a few repetitions, training loss plateaued, underscoring the vanilla RNN's shortcomings in handling dependencies.
Future Work: Architecture Upgrades: For improved sequence handling and dependency management, switch to more sophisticated architectures like LSTM or GRU.



A training loss history with a distinctive exponential decay pattern is seen on the graph. The first 20 iterations reveal a steep initial decline, followed by a gradual stabilization, from a high loss value of about 40. After about 60 rounds, the curve finally flattens out close to zero loss, demonstrating that the model has converged during training.