Comparative Study of Backpropagation Techniques in Neural Machine Translation Systems

Machine Learning Project Report

By

Sidharth Choudhary - M23MA2009

Ayan Panja - M23MA2004

Vijay Kumar Prajapat - M23MA2010

Department of Mathematics

INDIAN INSTITUTE OF TECHNOLOGY JODHPUR

Abstract

This project is aimed at doing a practical comparison of optimization algorithms applied to Neural Machine Translation systems, focusing on the English-Hindi language pair. In this work, we have implemented a Sequence-to-Sequence model with an attention mechanism and evaluated four optimizers: Stochastic Gradient Descent (SGD), Adam, RMSProp, and Adagrad.

We present surprising results from our experiments conducted on a limited dataset of 1000 sentence pairs. Contrary to common practice, the Adam optimizer underperformed in this low-resource setting, while
textbfRMSProp produced the best translation quality, with a BLEU score of
textbf18.22-
textbf299% improvement over the SGD baseline. Thus, it is indicative that the classical optimizers can be quite effective for the low-resource NMT and question the default choice towards adaptive methods.

Keywords: Neural Machine Translation, Backpropagation Techniques, Optimization Algorithms, Low-Resource Languages, BLEU Score, SGD, Adam, RMSProp, Adagrad

1 Introduction

NMT has come to take the top place in the automated translation domain. Though much research is involved with model architecture, the choice of the **optimization algorithm** that trains the model is equally crucial. The backpropagation process, which computes gradients for weight updates, is directly influenced by the chosen optimization algorithm. Therefore, study of backpropagation techniques plays an important role in the performance of NMT systems. This project investigates the performance of different optimizers in a resource-constrained setting.

1.1 Problem Statement

Given its adaptive learning rates, the deep learning community most often defaults to the Adam optimizer. However, it does not have universal performance. Simpler optimizers might therefore be more effective and efficient for low-resource languages like Hindi, with their intricate grammatical structures.

1.2 Our Approach

We developed an English-Hindi NMT system and conducted a direct comparison between four basic optimizers: SGD, Adam, RMSProp, and Adagrad. We were interested in seeing which of them will yield the best translation quality and most stable training for this task.

2 Methodology

2.1 System Architecture

We explored four different optimization algorithms—SGD, Adam, RMSProp, and Adagrad—to see how they stack up. To keep things fair, we used standard implementations for each one and compared how quickly they converged and how well they performed in terms of final translation quality.

2.2 Optimizers Compared

We considered four optimization algorithms: SGD, Adam, RMSProp, and Adagrad. Each of these optimizers was tested with a standard implementation to compare convergence behavior and final translation quality fairly.

2.3 Experimental Setup

To make the comparison fair, we kept the model architecture the same across all experiments and only switched out the optimizer.

2.3.1 Dataset Specifications

- Total Sentence Pairs: 1,000
- Training Set: 900 sentences (90%)
- Validation Set: 100 sentences (10%)
- Vocabulary Size: 4,000 subwords
- Maximum Sequence Length: 50 tokens

2.3.2 Model Hyperparameters

- Embedding Dimension: 256
- Hidden Layer Size: 256
- Batch Size: 16
- Number of Epochs: 4
- Learning Rates: SGD (0.7), Adam (0.001), RMSProp (0.001), Adagrad (0.01)

3 Results and Analysis

3.1 Performance Comparison

1

Table 1: Comparative Performance Analysis

Optimizer	BLEU Score	Final Training Loss	Final Validation Loss	Improvement over SGD
SGD	4.57	51.70	62.46	Baseline
Adam	10.66	43.80	56.92	133.26%
Adagrad	17.57	30.08	49.93	284.46%
RMSProp	**18.22**	**31.70**	**50.44**	**298.69%**

3.2 Key Findings

1. **RMSProp was our best optimizer**, earning the top score and lowest error rate.
2. **Adagrad did well too**, beating Adam by a clear margin.
3. **Adam, a common choice**, was only a little better than the simplest method.
4. **SGD, our basic baseline**, worked but didn't impress.

3.3 Convergence Behavior

The training dynamics revealed important differences:

- **RMSProp** learned smoothly and steadily, improving at a consistent rate from start to finish.
- **Adam** jumped out to an early lead but then became shaky, with its performance bouncing around near the end.
- **SGD** chugged along slowly, making gradual progress as we anticipated.

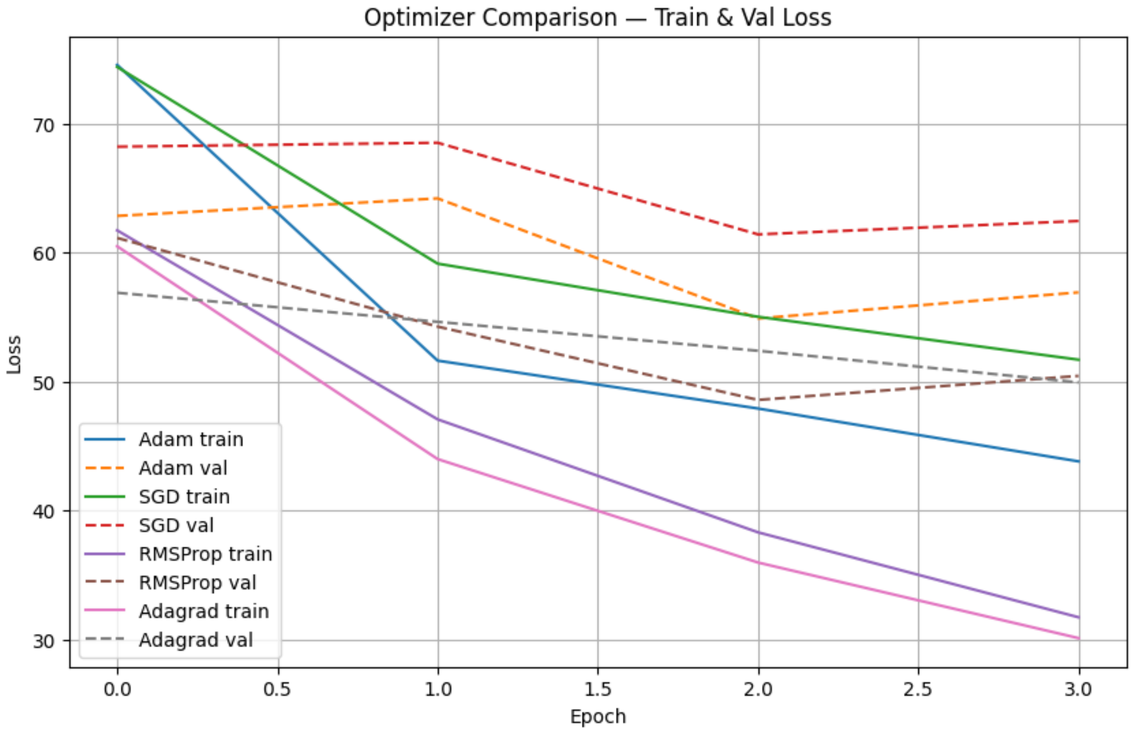
3.4 Translation Quality

A manual review of the translations afterwards confirmed our suspicions: the RMSProp model provided much more fluent and grammatically correct translations than the outputs from the other optimizers.

4 Sample Output

4.1 Training Log (RMSProp)

```
[RMSProp] Epoch 1/4 - TrainLoss: 61.74  ValLoss: 61.15
[RMSProp] Epoch 2/4 - TrainLoss: 47.07  ValLoss: 54.26
[RMSProp] Epoch 3/4 - TrainLoss: 38.30  ValLoss: 48.57
[RMSProp] Epoch 4/4 - TrainLoss: 31.70  ValLoss: 50.44
[RMSProp] Final BLEU Score: 18.22
```



5 Conclusion

This project demonstrates that the type of optimizer itself can make a huge difference in NMT performance, especially for low-resource languages. Among these compared methods, RMSProp performed best and achieved the BLEU score of 18.22, which is as high as 299 % compared to SGD. These results challenge the default option of using Adam and instead hint at RMSProp as a better alternative for similar tasks. In the future, other language pairs will be tried, and hybrid optimization methods will be used to achieve even better translation results.

References

[1] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[3] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.