Evaluating model performance on small datasets

Nam Anh Tran
Artificial Intelligence Institute
University of Engineering and Technology, VNU
Ha Noi
22022569@vnu.edu.vn

Abstract

In this paper, we introduce two novel simplified models based on QAnet and BiDAF, named S-QAnet and S-BiDAF, respectively. These models are designed to maintain the core strengths of their original architectures while reducing computational complexity. Additionally, we explore and compare various strategies for fine-tuning pre-trained language models on specific tasks. Our analysis delves into the impact of different fine-tuning techniques on model accuracy, generalization, and training stability.

1 Introduction

Deep learning has revolutionized various natural language processing (NLP) tasks, including Question Answering (QA), Natural Language Inference (NLI), and Machine Translation (MT). These tasks, each fundamental to understanding and generating human language, have seen significant advancements due to the development and application of deep learning models

Question Answering (QA) involves developing systems that can answer questions posed in natural language. QA systems are integral to various applications, from search engines to virtual assistants, and are essential for enabling machines to interact with humans in a meaningful way. The advent of deep learning and a new simple network architecture, the Transformer, particularly models such as BERT (Bidirectional Encoder Representations from Transformers) , has drastically improved the accuracy and efficiency of QA systems by enabling better contextual understanding and response generation.

Machine Translation (MT) involves automatically converting text from one language to another. MT is essential for bridging communication barriers in our increasingly globalized world. Traditional rule-based and statistical methods of translation have been largely outperformed by neural machine translation (NMT) models. NMT leverages deep learning techniques to provide more accurate and contextually appropriate translations by learning from vast amounts of bilingual text data. Notable advancements include the development of the Transformer model, which has become the backbone of state-of-the-art NMT systems due to its ability to handle long-range dependencies and parallelize training processes.

In this paper, we introduce two new architectures, S-BiDAF and S-QAnet, hierarchical multi-stage architecture modified versions of BiDAF and QAnet, along with which we evaluate the performance of S-BiDAF models, S-QAnet, RNN, and BERT in the QNLI task, and RNN, Transformer, VinAI translate models in the MT task.

2 Methods

In this section, the methods utilized for the Question-answering NLI task and the Machine Translation task are presented in this section.

Preprint. Under review.

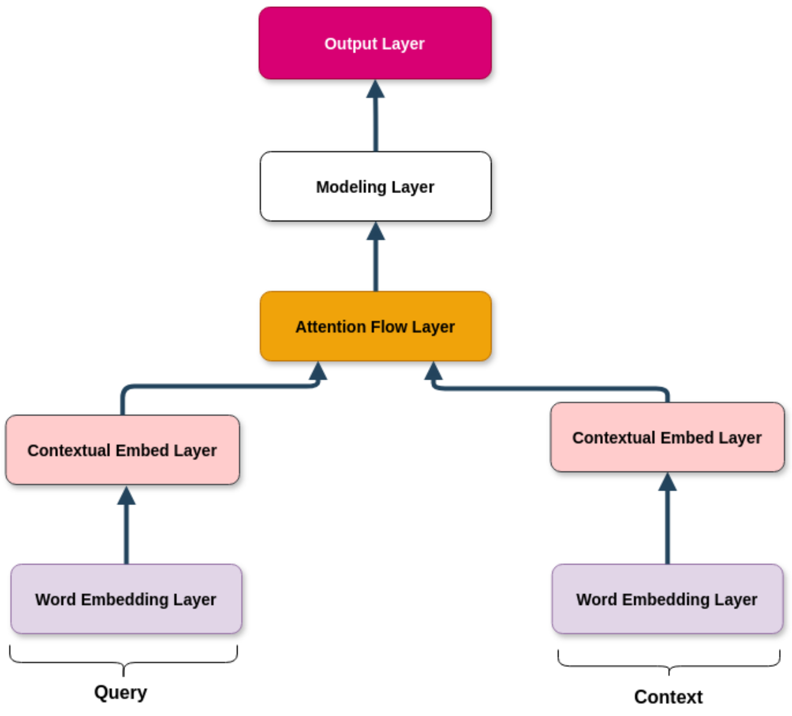

Figure 1: S-BiDAF

2.1 Question-answering NLI

This section presents the problem that needs to be solved first, followed by the suggested models are S-QAnet, BERT, RNN, and S-BiDAF.

2.1.1 Problem Formulation

The question-answering task considered in this paper is defined as follows: Given a context paragraph C and the query sentence Q as input, $A = \{0, 1\}$ as output, which is the answer whether or not the context paragraph C contains the answer to the query sentence Q, 1 if yes, otherwise is 0.

2.1.2 S-BiDAF

1. Word Embedding Layer. To encode each word into its embedding vector, we employ the word embedding model. In this research, we tested two distinct word embedding models. I trained the word embedding model from scratch in the first experiment. The pre-trained BERT word embedding model is used for the second word embedding model; however, the word embedding weight of BERT is frozen in its entirety.

2. Contextual Embedding Layer. This layer contextualizes the words in the query sentence and context paragraph by taking the output from the word embedding layer as input to the bidirectional RNN model. So that each word in a query sentence or context paragraph not only has its own meaning, but the surrounding words can also affect the meaning of that word itself.

3. Attention Flow Layer. This layer is the same as the Attention Flow Layer in [1] . After the words are in context paragraph C and the words are in query sentence Q, they go through the contextual embedding layer. This layer takes output from the contextual embedding layer, including encoded context and encoded queries, as input for the Attention Flow Layer. The encoded context vector C^e and encoded query vector Q^e are first used to calculate the similarity matrix S, which represents the similarity level between the encoded context vector and the encoded query vector.

Context-to-query Attention (C2Q) C2Q signifies which query words are most relevant to each word in the context paragraph. Firstly, the attention weight C^Q is vector S that is normalized by applying $softmax$ on each row, this vector that rank the degree of similarity between words in context with words in the query sentence, . Vector C^Q then is used to calculate the attended query vector H, which is the representation of each word in context paragraph by words in the question sentence.

Query-to-context Attention (Q2C) Q2C signifies which context words have a closet similar to one of the query words. Firstly, calculating the attention weight vector Q^C, then use Q^C to compute the attended context vector U, which is the representation of each word in question sentence by words in context paragraph.

Finally, the contextual embeddings and the attention vectors are combined together to yield G, where each column vector can be considered as the query-aware representation of each context word

4. Modeling Layer. We use the query-aware representation of context word as the input of the modeling layer, which is the bidirectional recurrent neural network. This layer capture the interaction or relevant between each context word and query-aware context words that is different from the contextual embedding layer, which is only capture the relevant between context and query independently

5. Output Layer. The output layer is task-specific. It consists of two fully connected neural network layers; its task is to synthesize important information from which to decide whether the context paragraph contains the answer to the query sentence or not.

2.1.3 S-QAnet

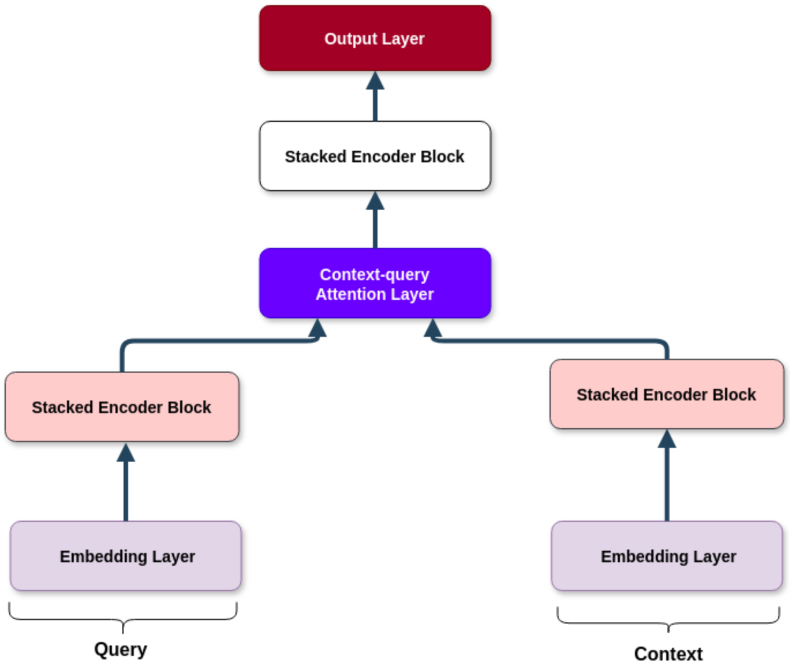

Figure 2: S-QAnet

2.1.4 S-QAnet

1. Embedding Layer. The embedding layer consists of a word embedding model and positional encoding. It takes input as the index of the word and then outputs its embedding vector plus position embedding, which is the word itself that contains its meaning and positional information.

2. Stacked Encoder Block. The Stacked Encoder Block can be called Contextual Embed Layer, which includes many Transformer-Encoder layers that help embed vectors from the Embedding Layer to have a fuller meaning. The embedding vector not only includes the meaning of the word itself, but the meaning of the words also depends on the meanings of surrounding words, helping the words become more meaningful in the sentence, by taking advantage of the self-attention mechanism.

3. Context-query Attention Layer. Context-query Attention Layer layer is almost the same as the Attention Flow Layer in [1]. It first compute the similarities matrix between each pair of context and query words. Then it using this matrix to compute the Context-to-query attention and Query-to-context attention, and combine them, which is query-aware context and context-aware query, with context vector to produce a matrix F.

4. Stacked Encoder Block The Stacked Encoder Block is used the same as the ones above, but it synthesize relevant from both context vector and query-aware context and context-aware query.

5. Output Layer The output layer is task-specific and its task is the same as the Output Layer in S-BiDAF. Its task is output whether the context contain the answer to the query.

2.1.5 RNN

We use the Sequence to Sequence model [2], with the Encoder take the embed word from question sentence as input and then produce a hidden vector, which is the vector that compress any important information and then pass it the Decoder layer. The Decoder layer take the hidden vector and embed word from context paragraph as input. Its task is compute the similarities between each word with hidden vector. Afterwards its output then pass into the fully connected neural network to synthesize important information and then make decision whether or not the context contain the answer to the query.

2.1.6 Fine-tuning BERT

We used the BERT [3] base model to fine-tune the QNLI task, in which we experimented with two different fine-tuning methods.

Method 1: Fine-tuning the embedding layer and classifier. We experiment with only BERT-base architecture [3]

Method 2: Fine-tuning the entire model. We experiment with only BERT-base architecture [3]

Method 3: Freezing top-k layers [4]. We experiment with BERT-base architecture and BERT-large architecture [3]. We decide to fine-tune while freezing the first k layers since prior research [4] has demonstrated that instability frequently arises during fine-tuning the first k layers.

Method 4: Fine-tuning with the number of iterations is as many as the number of iterations in a large dataset [4]. We experiment with BERT-large architecture. We decide to fine-tune model with almost the same number of iterations as fine-tuning on large dataset, because previous work had proved that training on less data does indeed affect the fine-tuning variance, in particular, there are many more failed runs. However, when we simply train for as many iterations as on the full training dataset, we almost completely recover the original variance of the fine-tuning performance [4].

2.2 Machine Translation

This section presents the problem that needs to be solved first, followed by the suggested models are RNN, Transformer, VinAI Translate.

2.3 Problem Formulation

The machine translation task is considered in this work is defined as follows. Given an English sentence as input, output a Vietnamese sentence.

2.3.1 RNN

We use the same Sequence to Sequence model as in [2], with the Encoder takes an English sentence as input. The Decoder takes the hidden vector from the Encoder and its input, which is then used to predict the Vietnamese sentence.

2.3.2 Transformer

We use the same Transformer model as in [5].

2.3.3 VinAI translate [6]

In this work, we will use VinAI translate [7] model, to experiment with different fine-tuning strategy.

Method 1: Fine tuning not using any template.

Method 2: Fine tuning using template "Translate the following [src] sentence into [tgt]:[input]" [8]

3 Experiments

3.1 Data

In this work, we use two separate datasets for each task: IWSLT'15 English-Vietnamese [9] for the machine translation task and the Stanford Question Answering Dataset [10] for the QNLI task. For the machine translation task, we selected 4096 samples for the training dataset, 512 samples for the development dataset, and 256 samples for the testing dataset. As for the QNLI task, we selected 10000 samples for the training dataset with 5013 questions with answers in sentences and 4987 samples with no answers in sentences, 1000 samples for the development dataset, and 1000 samples for the testing dataset.

3.2 Settings

In the QNLI task, we use the accuracy metric to compare the performance of trained S-BiDAF, S-QAnet, and RNN models with fine-tuned BERT-base and BERT-large models on our selected QNLI dataset. For the models evaluated in the QNLI task, we use AdamW [11], ADADELTA [12], or Adam [13] Optimizer to train our models. Along with that, we also use a linear learning rate scheduler with a warm-up step.

For the machine translation task, we use the BLEU [14] score metric to evaluate the trained RNN and Transformer models, along with the fine-tuned VinAI model, on our selected MT dataset. We also use AdamW [11] to train our models. Along with that, we also use a linear learning rate scheduler with a warm-up step.

3.3 Results

In the QNLI task, after performing fine-tuning and training the model with different parameter sets in BiDAF [1] and QAnet [15], we extract the best results of the model from the test set as shown in the Table 1. For the MT task, we obtained the results as shown in the Table 2. In which the BERT-large model achieved the best results in the QNLI task and VinAI [7] achieved the best results in the MT task. We also provide the results of testing the BERT-base model with different fine-tune methods in Table 3. Not only that, we also fine-tune BERT-large, since previous experiments demonstrated that the BERT-base model is unstable the Table [4]. With the fine-tuning method, we tested freeze top-k layers because previous research has demonstrated instability when fine-tuning BERT that often occurs when fine-tuning the top-k layers [4]. In this experiment, we chose $k = 12$, because previous work showed that $k = 12$ [4] is one of the values of k for which the model is stable when it is fine-tuned. We also experimented with different optimizers, learning rates, and number of epochs, as in [4], obtaining the results in Table 4.

Table 1: Model performance in QNLI Task

Model name	Baseline	RNN	S-BiDAF	S-QAnet	BERT-base	BERT-large
Accuracy	50%	52%	51%	52%	85%	**89%**

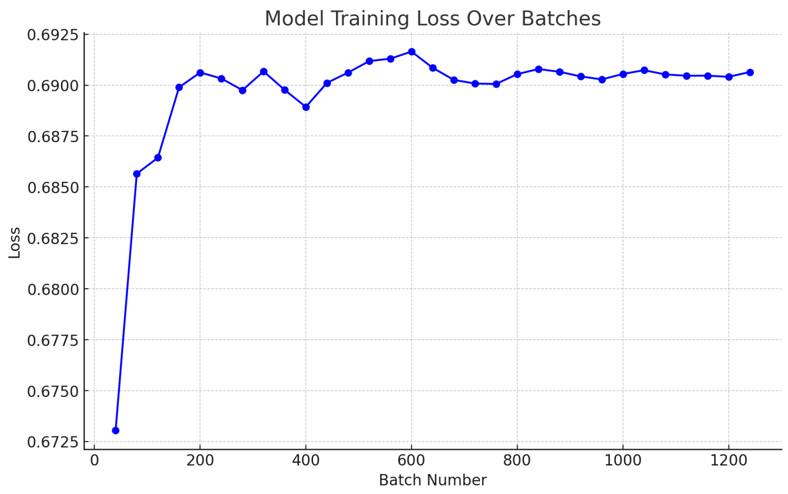

Figure 3: Failed fine-tuning.

3.4 Analyses and Discussions

After fine-tuning and training the above models many times, we have made a few observations. First, models with failed fine-tuning results often have loss values that increase or fluctuate around a loss value within a very small range, as shown in Figures **??** and 5. Second, for models with successful fine-tuning results, the loss value often decreases very quickly in the initial few epochs, as shown in Figure 4. Therefore, when fine-tuning the models, we will consider if the models have If the phenomenon looks like Figure 3 or 5, we will change the parameter set now. Third, if the learning rate value is too large, there are some cases where, although the loss value tends to decrease, like in Figure 4, as the last epochs get closer, the loss value tends to fluctuate very strongly, or When the loss value starts to be low, and tends to fluctuate very strongly, as shown in Figure 6, we will decide to reduce the learning rate. We also provide the loss value for training RNN in the QNLI task in the Figure 7. Although the model training process obtained good results, we suspect that because the data set is small, the generalization is still low. so it does not achieve high performance in the evaluation step.

3.5 Limitations

We have not been able to thoroughly examine our model when trained on huge datasets because of hardware restrictions throughout the training process. We have only used a few of the techniques described in the study due to time restrictions. In the future, we hope to combine cutting-edge models with a wide variety of additional fine-tuning techniques.

Table 2: Model performance in MT Task

Model name	RNN	Transformer	VinAI
BLEU	0.62	0.15	**58.9**

Table 3: Results after BERT-base is fine-tuned

Method	Classifier + Head	Entire	Freeze top-k layers
Accuracy	58%	**88%**	84%

Table 4: Results of BERT-large when the top 12 layers are frozen.

Model	Optimizer	Learning rate	Epochs	Accuracy
BERT-large	$AdamW$	$5e-5$	3	50%
	$Adam$	$5e-5$	3	50%
	$AdamW$	$1e-5$	3	87%
	$AdamW$	$1e-5$	10	**88%**

3.6 Future Work

We want to widely deploy new methods such as Continued Pre-Training [16], Task Adaptive Pretraining [16], Prompt-based Continued Pre-training [16], Prompted Masked Language Modeling [16]. Perform fine-tuning on larger models like Llama 3, Mistral [17] along with methods like Parameter Efficient Fine Tuning [18].

4 Conclusion

After training three models, S-BiDAF, S-QAnet, and RNN, and fine-tuning BERT in two different ways, In the QNLI task, for S-BiDAF, S-QAnet, and RNN, we show that the difference in performance of the models is very small, almost negligible when trained on small data sets, only at a performance level equal to the baseline. For fine-tuning BERT using different methods, we found that fine-tuning only the embedding layer and classifier layer achieved significantly worse performance than fine-tuning the entire model or fine-tuning with top-k layers is frozen. In particular, with fine-tuning, the entire model achieves a performance increase of nearly 51.72% compared to fine-tune embedding layer and classifier layer. For the machine translation task, we found that for small data sets, RNN seems to give better results than Transformer. However, when testing translating text from an English sentence to a Vietnamese sentence, both model are worse. For the machine translation task, we found that for small datasets, RNN seems to give better results than Transformer, but both models give very low BLEU scores. In the QNLI task, although the dataset is small, BERT-large and BERT-base both give very good results. As for the MT task, for small data sets, RNN and Transformer give poor translation results, but VinAI still gives good results.

References

[1] Seo, M., A. Kembhavi, A. Farhadi, et al. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[2] Bahdanau, D., K. Cho, Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Devlin, J., M.-W. Chang, K. Lee, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[4] Mosbach, M., M. Andriushchenko, D. Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*, 2020.

[5] Vaswani, A., N. Shazeer, N. Parmar, et al. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Liu, Y., J. Gu, N. Goyal, et al. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[7] Nguyen, T.-D. H., D. Phung, D. T.-C. Nguyen, et al. A vietnamese-english neural machine translation system. In *INTERSPEECH*, pages 5543–5544. 2022.

[8] Zheng, J., H. Hong, X. Wang, et al. Fine-tuning large language models for domain-specific machine translation. *arXiv preprint arXiv:2402.15061*, 2024.

[9] Luong, M.-T., C. D. Manning. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*. Da Nang, Vietnam, 2015.

[10] Rajpurkar, P., J. Zhang, K. Lopyrev, et al. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, pages 2383–2392. Association for Computational Linguistics, 2016.

[11] Loshchilov, I., F. Hutter. Decoupled weight decay regularization, 2019.

[12] Zeiler, M. D. Adadelta: An adaptive learning rate method, 2012.

[13] Kingma, D. P., J. Ba. Adam: A method for stochastic optimization, 2017.

[14] Papineni, K., S. Roukos, T. Ward, et al. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. 2002.

[15] Yu, A. W., D. Dohan, M.-T. Luong, et al. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[16] Gururangan, S., A. Marasović, S. Swayamdipta, et al. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[17] Jiang, A. Q., A. Sablayrolles, A. Mensch, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[18] Hu, E. J., Y. Shen, P. Wallis, et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

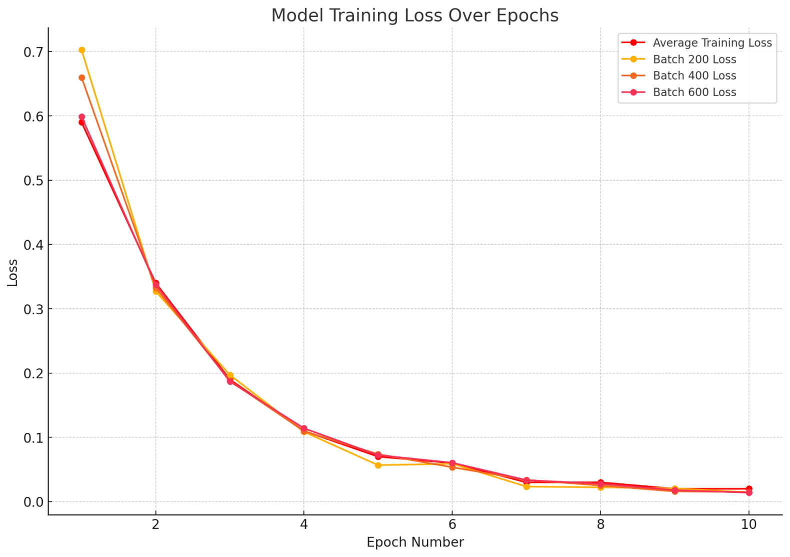

Figure 4: Successfully fine-tuning.

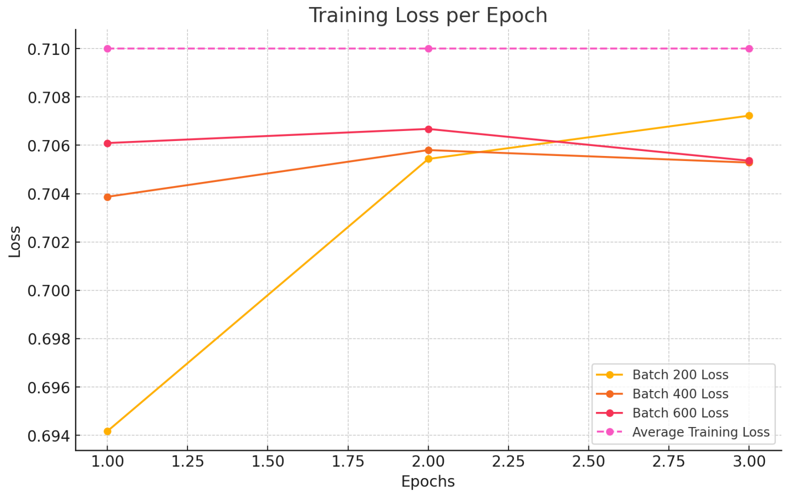

Figure 5: Failed fine-tuning BERT.

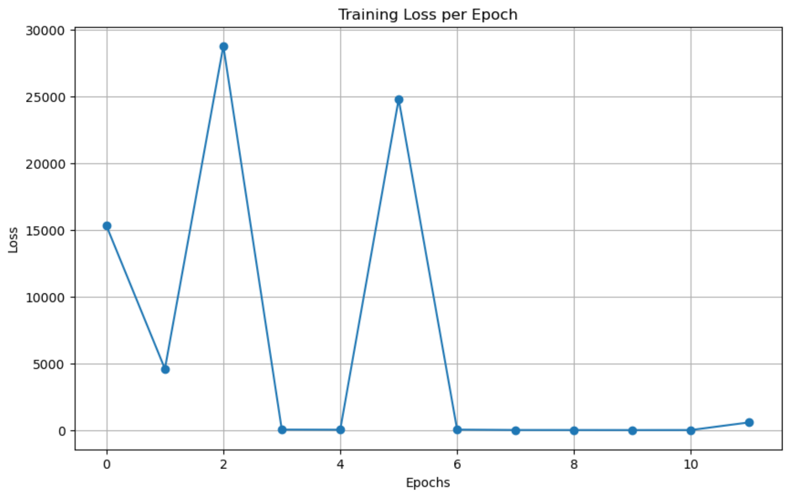

Figure 6: Failed due to learning rate too high.

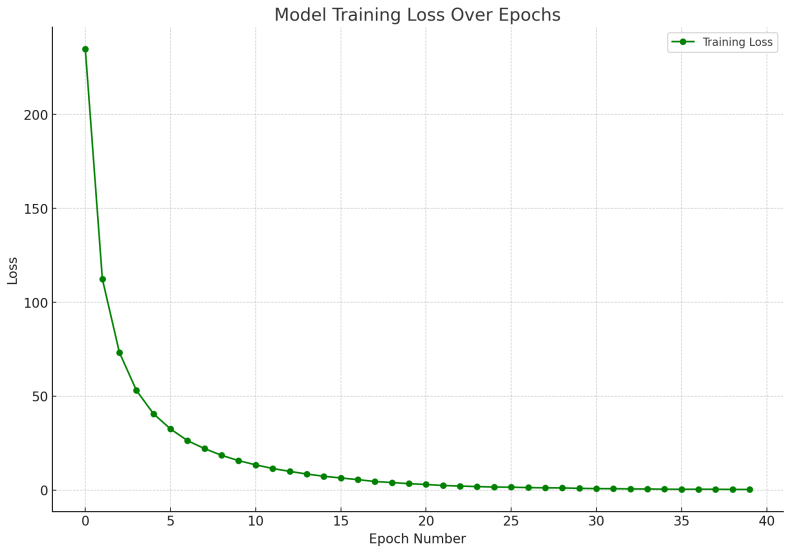

Figure 7: RNN Training Loss.