

# Accelerating Bangla NLP Tasks with Automatic Mixed Precision: Resource-Efficient Training Preserving Model Efficacy

Md Mehrab Hossain Opi

*Department of Computer Science and Engineering  
Khulna University of Engineering & Technology  
Khulna, Bangladesh  
Email: opi@cse.kuet.ac.bd*

Sumaiya Khan

*Department of Computer Science and Engineering  
Khulna University of Engineering & Technology  
Khulna, Bangladesh  
Email: sumaiyakhan3572213@gmail.com*

Moshammad Farzana Rahman

*Department of Computer Science and Engineering  
Khulna University of Engineering & Technology  
Khulna, Bangladesh  
Email: rfarzana964@gmail.com*

**Abstract**—Training models for Natural Language Processing (NLP) requires substantial computational resources and time, posing significant challenges, especially for NLP development in Bangla, where access to high-end hardware is often limited. In this work, we explore automatic mixed precision (AMP) training as a means to improve computational efficiency without sacrificing model performance. By leveraging a dynamic mix of 16-bit and 32-bit floating-point computations, AMP lowers GPU memory requirements and speeds up training without degrading model performance. We evaluate AMP across four standard Bangla NLP tasks, namely sentiment analysis, named entity recognition, error classification, and question answering, using four transformer-based models: BanglaBERT, BanglBERT, XLM-R, and mBERT. Our results demonstrate that AMP accelerates training by 44.5% and reduces memory consumption by 17.6%, while maintaining F-1 score within 99.7% of the full-precision baselines. This empirical study highlights AMP’s potential to democratize access to state-of-the-art NLP capabilities in hardware-constrained settings by lowering computational barriers.

**Index Terms**—HPC, NLP, AMP, Benchmarking

## I. INTRODUCTION

Natural Language Processing (NLP) has become a foundational subfield of artificial intelligence, enabling applications such as machine translation, sentiment analysis, and conversational agents, and has emerged as a core research area powering systems from search engines to intelligent virtual assistants [1]. Recent breakthroughs in Large Language Models (LLMs) have catalyzed further progress, demonstrating unprecedented task performance across multiple NLP domains [2]. Despite notable progress, research has largely centered on high-resource languages, leaving Bangla, one of the world’s most widely spoken languages, comparatively underrepresented. Emerging pretrained Bangla language models [3]–[6] have demonstrated strong potential on downstream tasks. However, the training and deployment of large-scale

NLP models are computationally demanding, necessitating substantial memory and processing capacity on modern GPUs. These requirements pose significant challenges for researchers and organizations working with limited hardware resources, a situation common in developing regions where Bangla NLP research is gaining momentum, emphasizing the urgent need for efficiency-oriented solutions.

Mixed-precision training [7] has emerged as an effective approach to address these computational challenges. It leverages a combination of 16-bit floating-point (FP16) and 32-bit floating-point (FP32) data types during training. By performing most computations in the smaller FP16 format, mixed-precision training effectively halves memory usage and can significantly accelerate matrix multiplication operations, which are the backbone of modern deep learning. This enables models to leverage lower-precision arithmetic without compromising accuracy. While AMP has demonstrated substantial improvements in training speed and memory efficiency for high-resource language models, to the best of our knowledge, no systematic study has yet investigated its impact on Bangla NLP tasks. This leaves a critical gap in understanding its effectiveness in this important yet under-explored context, especially for overcoming the hardware limitations faced by local researchers.

In this work, we investigate the effectiveness of Automatic Mixed Precision for Bangla NLP tasks using GPU-based training. We select a few widely used pretrained Bangla language models as the baseline and apply AMP to accelerate training while reducing memory overhead. The objective is to evaluate whether AMP can deliver measurable improvements in training efficiency, specifically in training time and GPU memory utilization without degrading task performance. To achieve this, we conduct experiments on four standard Bangla NLP tasks using widely-used Transformer-based models. We sys-

tematically compare full-precision and AMP-enabled training across multiple performance metrics. Our findings demonstrate that AMP significantly reduces training time and memory consumption while maintaining model accuracy.

The main contributions of this work are as follows:

- We implement automatic mixed precision (AMP) training on multiple pretrained Transformer models for Bangla NLP tasks.
- We systematically evaluate AMP across four representative Bangla NLP tasks to quantify efficiency and performance trade-offs.
- We demonstrate substantial efficiency gains with AMP, including reduced GPU memory usage and faster training, while maintaining model performance.
- We provide practical guidance for optimizing training workflows in low-resource Bangla NLP settings without compromising accuracy.

## II. RELATED WORK

Research in Bangla NLP has advanced considerably with the development of pretrained language models tailored to the language. BERT-based architectures, such as BanglaBERT and BanglighBERT [3], have been successfully applied to various NLP tasks, trained on large Bangla corpora. Recent encoder-decoder models, including BanglaT5 [4] and BanglaByT5 [5], facilitate low-resource natural language generation in Bangla, while BanglaGPT [6] demonstrates generative pretrained transformer capabilities. Multilingual transformers such as mBERT [8], MuRIL [9], and XLM-RoBERTa (XLM-R) [10] achieve competitive performance for Bangla despite being trained on multiple languages. More recently, Bangla-focused large language models (LLMs), including BongLLaMA [11] and TigerLLM [12], indicate growing efforts to bring LLM capabilities to Bangla NLP. Despite their strong performance, these models require substantial GPU resources, limiting accessibility in low-resource settings.

Mixed-precision training [7] has emerged as a widely adopted technique for accelerating deep learning by combining lower-precision arithmetic (e.g., FP16) with standard precision (FP32). Automatic Mixed Precision (AMP), available in frameworks such as PyTorch and TensorFlow, improves training speed and reduces memory consumption without degrading model accuracy in high-resource language tasks [13], [14]. While AMP has been applied to large-scale NLP models in English and other widely studied languages, its potential in Bangla NLP remains unexplored.

In summary, although pretrained Bangla models exist and AMP has proven effective in other languages, systematic studies applying AMP to Bangla NLP tasks are missing. Our work addresses this gap by evaluating AMP across multiple Bangla NLP benchmarks, measuring both task performance and training efficiency, thereby providing practical insights for researchers working with limited computational resources.

## III. METHODOLOGY

This study investigates the efficiency of Automatic Mixed Precision (AMP) training across multiple Bangla NLP tasks using pretrained transformer architectures. We design a unified training and evaluation pipeline to enable consistent comparison between full precision (FP32) and mixed precision (FP16/FP32) training. Four representative tasks are selected to cover sentence-level, token-level, and span-level NLP problems. Pretrained monolingual and multilingual transformer models—including BanglaBERT [3], BanglighBERT [3], mBERT [8], and XLM-R [10]—are fine-tuned under both training regimes. Performance is evaluated using accuracy- and efficiency-oriented metrics, highlighting the balance between computational resource usage and task performance.

### A. Task Description and Dataset Overview

We evaluate AMP training on four Bangla NLP tasks:

- **Sentiment Analysis (SA):** Classifies text into positive, negative, or neutral sentiment using the BLP-2023 [15] dataset, which integrates MUBASE [16] and SentiNob [17], comprising 46,643 examples. Preprocessing includes removal of URLs, emojis, and special characters, followed by model-specific tokenization using Sentence-Piece.
- **Named Entity Recognition (NER):** Identifies named entities in Bangla health-related text using the Bangla-HealthNER dataset [18], containing 31,783 annotated samples with seven entity types in IOB format. The dataset includes code-switched Bangla-English text.
- **Error Classification (EC):** Detects and categorizes errors in user-generated Bangla text using the BaTECor [19] dataset, comprising 10,000 YouTube comments across multiple domains. Only comments with at least three Bangla words are included.
- **Question Answering (QA):** Answers questions based on Bangla passages using the BanglaRQA dataset [20], containing 14,889 question-answer pairs over 3,000 Wikipedia passages. Questions are of four types (list, factoid, causal, confirmation) and answers of three types (yes/no, single span, multiple spans).

Preprocessing for all tasks includes normalization, tokenization, and truncation/padding to fit model input requirements.

### B. Model Selection

Four pretrained transformer models are selected to represent monolingual and multilingual approaches:

- **BanglaBERT:** BERT-based model pretrained on Bangla text [3].
- **BanglighBERT:** Trained on code-mixed Bangla-English text to capture social media linguistic nuances [3].
- **mBERT:** Multilingual BERT pretrained on 104 languages including Bangla, offering cross-lingual capabilities [8].
- **XLM-R:** Multilingual transformer pretrained on 100+ languages, known for robust cross-lingual transfer [10].

### C. Methodology Pipeline

The training pipeline, illustrated in Figure 1, follows these steps:

- 1) **Task Selection:** Four representative Bangla NLP tasks are selected to evaluate AMP training.
- 2) **Data Preprocessing:** Text is normalized using the normalizer library, tokenized with model-specific tokenizers, and padded to maximum sequence lengths with attention masks.
- 3) **Model Initialization:** Pretrained transformers are augmented with task-specific classification or span prediction heads; token-level classification is used for NER and EC tasks.
- 4) **Training Setup:** Models are trained under FP32 and AMP using AdamW with task-specific learning rates and weight decay. Gradient clipping and early stopping are applied based on validation performance. Batch sizes and maximum sequence lengths are given in Table I.
- 5) **Evaluation:** Task performance is measured using Accuracy, F1-score, and Exact Match (EM), while efficiency is assessed via peak GPU memory, epoch time, and throughput.
- 6) **Comparison and Analysis:** FP32 and AMP results are systematically compared to quantify trade-offs between model performance and computational efficiency.

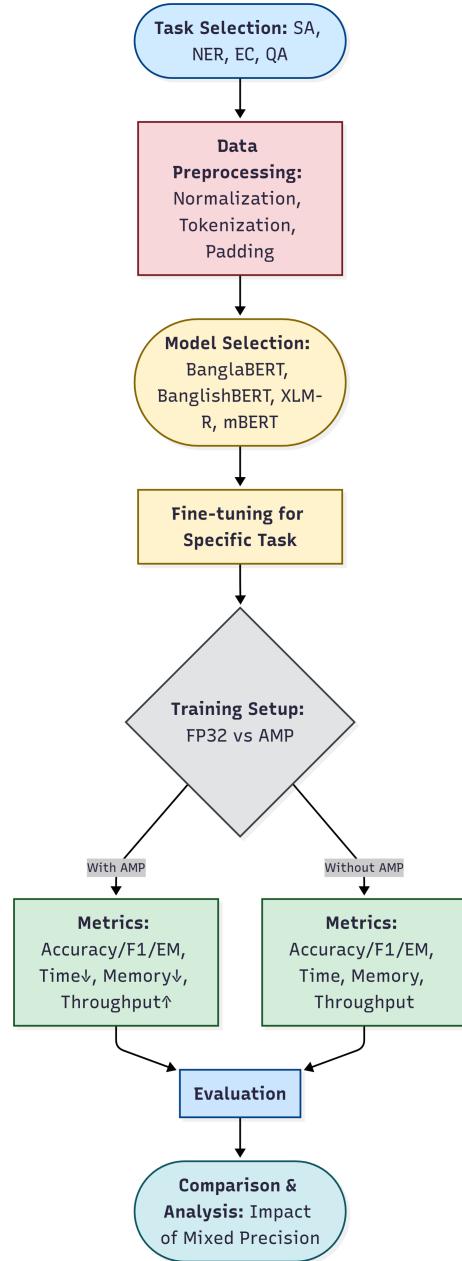
### D. Implementation Details

Experiments are implemented in PyTorch with the Hugging Face Transformers library. Token sequences are padded to model-specific maximum lengths with attention masks applied to distinguish real tokens from padding. Gradient clipping stabilizes training, and model checkpoints are saved at regular intervals. AMP training is facilitated via PyTorch’s `torch.cuda.amp` module, reducing memory usage and accelerating training while maintaining accuracy comparable to FP32.

All experiments are conducted on an Intel Core i5-13500 CPU with an NVIDIA RTX 4070 GPU and 64 GB DDR4 RAM running Ubuntu 22.04. Task-specific batch sizes, learning rates, and epochs are summarized in Table I.

## IV. EXPERIMENT AND RESULT

Our analysis investigates the impact of Automatic Mixed Precision (AMP) training on Bangla NLP tasks using pre-trained transformer models. We evaluate AMP across four representative Bangla NLP tasks: Sentiment Analysis, Named Entity Recognition (NER), Error Classification, and Question Answering, using pretrained transformer models including BanglaBERT, BanglBERT, mBERT, and XLM-R. We focus on two main aspects: (i) *task performance*, measured in terms of accuracy and F1-score for tasks of classification and sequence labeling, and Exact Match (EM) for Question Answering, and (ii) *training efficiency*, evaluated through GPU memory consumption, training time, and throughput. We also examine the effect of varying batch sizes on performance. All experiments compare the baseline FP32 training setup



**Fig. 1:** Overview of the proposed AMP training methodology for Bangla NLP tasks.

against AMP-enabled training to highlight both the benefits and potential trade-offs.

### A. Task Performance

We evaluate the impact of Automatic Mixed Precision (AMP) training utilizing FP16/FP32 mixed-precision across four Bangla NLP tasks: Sentiment Analysis (SA), Named Entity Recognition (NER), Error Classification (EC), and Question Answering (QA). Pretrained transformer models, namely BanglaBERT, BanglBERT, mBERT, and XLM-R, are fine-tuned under standard FP32 and AMP (FP16/FP32) regimes. Task performance is measured using Accuracy and

**TABLE I:** Task-Specific Model and Training Configurations

Task	Optimizer	Weight Decay	Learning Rate	Batch Size	Epochs	Max Seq Len	Dropout
Sentiment Analysis	AdamW	0.01	5e-5	16	3	512	0.1
NER	AdamW	0.03	1e-4	64	20	512	0.4
Error Classification	AdamW	0.01	1e-5	16	5	128	0.1
Question Answering	AdamW	0.00	2e-5	8	15	512	0.1

F1-score for Sentiment Analysis, Named Entity Recognition, and Error Classification, and F1-score and Exact Match (EM) for Question Answering. Tables II–V summarize the results.

Across all tasks, training using Automatic Mixed Precision preserves or slightly improves model performance relative to the full-precision baseline. F1 retention is 98.41–99.68% for Sentiment Analysis, 94.34–99.95% for NER, 100.73–122.52% for Error Classification, and 99.73–105.97% for QA, with occasional improvements likely due to implicit regularization. These results confirm that AMP maintains task efficacy while enabling substantial computational efficiency. Key observations include:

- **Sentiment Analysis:** AMP maintains comparable Accuracy and F1 across all models, with BanglaBERT achieving the highest scores.
- **NER:** F1 scores under AMP remain stable, with XLM-R and mBERT demonstrating the highest consistency.
- **Error Classification:** Accuracy is preserved under AMP, while F1 shows modest improvements for some models, reflecting robust token-level classification.
- **Question Answering:** AMP yields comparable or slightly higher F1 and EM scores, confirming no degradation in span-level performance.

**TABLE II:** Sentiment Analysis Performance: FP32 vs AMP

Model	Accuracy (FP32 / AMP)	F1 (FP32 / AMP)
BanglaBERT	72.70 / 72.27	72.69 / 72.26
BanglighBERT	71.33 / 71.10	71.32 / 71.09
XLM-R	68.25 / 67.39	68.25 / 67.38
mBERT	66.34 / 65.28	66.34 / 65.27

**TABLE III:** NER Task Performance: FP32 vs AMP

Model	Accuracy (FP32 / AMP)	F1 (FP32 / AMP)
BanglaBERT	88.52 / 88.25	55.78 / 54.35
BanglighBERT	88.89 / 88.74	57.54 / 56.71
XLM-R	89.08 / 89.23	58.50 / 58.47
mBERT	87.99 / 87.24	55.03 / 51.90

**TABLE IV:** Error Classification Performance: FP32 vs AMP

Model	Accuracy (FP32 / AMP)	F1 (FP32 / AMP)
BanglaBERT	73.88 / 72.64	52.88 / 54.13
BanglighBERT	72.63 / 72.19	52.61 / 53.00
XLM-R	70.15 / 70.04	42.29 / 51.79
mBERT	67.86 / 67.36	50.57 / 51.17

**TABLE V:** Question Answering Performance: FP32 vs AMP

Model	F1 (FP32 / AMP)	EM (FP32 / AMP)
BanglaBERT	57.95 / 60.10	41.46 / 44.60
BanglighBERT	55.02 / 56.28	37.31 / 38.98
XLM-R	48.12 / 47.99	31.81 / 31.95
mBERT	45.74 / 48.49	31.07 / 33.02

Overall, AMP effectively maintains task efficacy across sentence-level, token-level, and span-level NLP tasks, while enabling substantial computational savings, which are analyzed further in the training efficiency section.

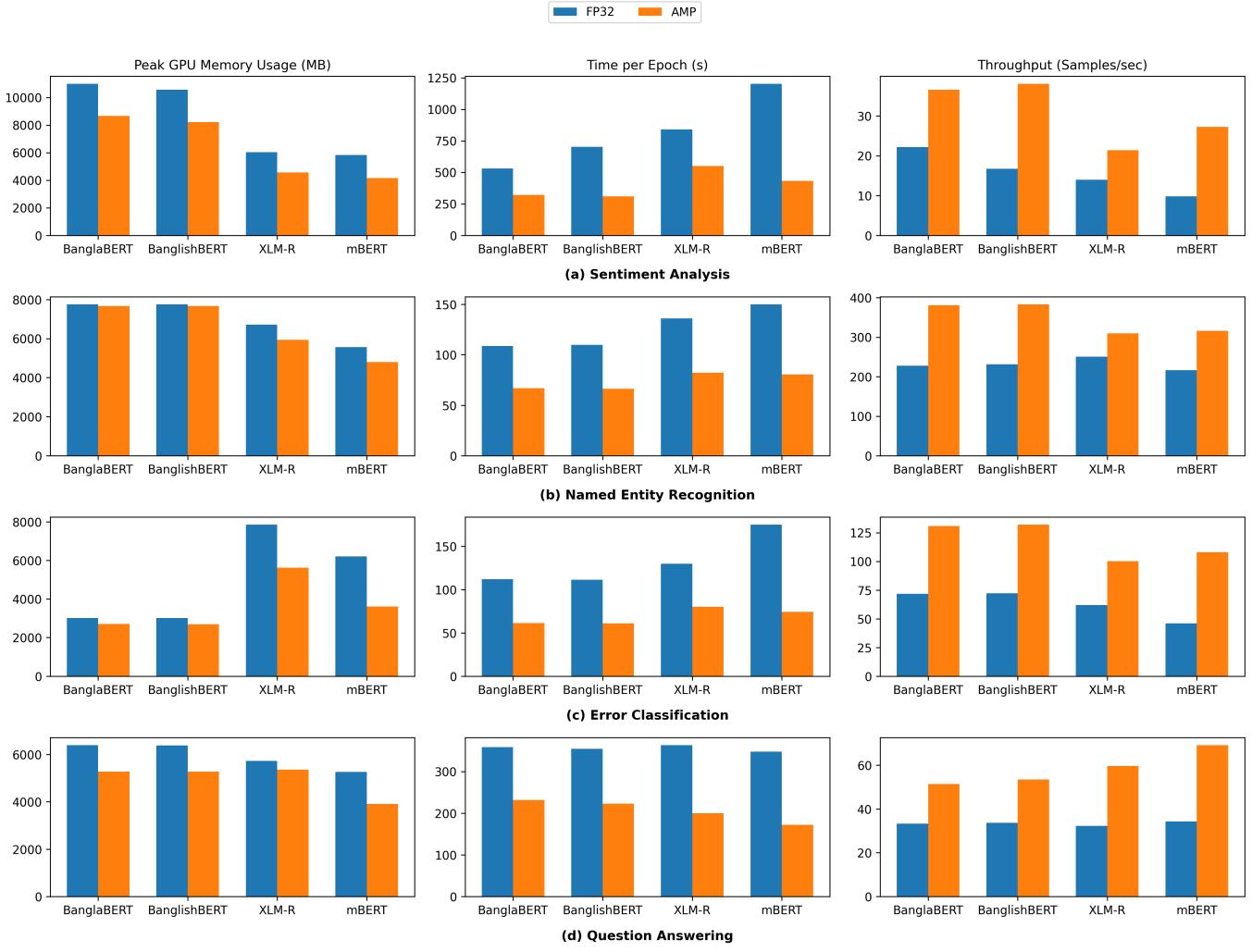
### B. Training Efficiency

We evaluate the computational efficiency gains of Automatic Mixed Precision (AMP) training (FP16/FP32) across four Bangla NLP tasks using pretrained transformer models. Three metrics are considered: (i) peak GPU memory consumption, (ii) throughput measured in samples per second, and (iii) training time per epoch. All experiments are conducted on identical hardware and software to ensure a fair comparison between FP32 and AMP. Results are summarized in Figure 2.

**Memory Usage.** Automatic Mixed Precision (AMP) consistently reduces GPU memory consumption across all models and tasks. The task-wise memory reductions range as follows: Sentiment Analysis: 1.2–21.2%, NER: 1.1–22.2%, Error Classification: 6.5–28.4%, QA: 13.9–41.9%. The largest savings are observed in larger multilingual models, such as XLM-R and mBERT, whereas smaller models, including BanglighBERT, show comparatively modest reductions. These reductions enable larger batch sizes and make training feasible in resource-constrained environments, highlighting AMP’s practical utility for Bangla NLP research.

**Throughput.** AMP significantly improves training throughput across all tasks and models. The observed task-wise increases in samples per second are as follows: Sentiment Analysis: 54–82%, NER: 59–127%, Error Classification: 23–85%, QA: 46–179%. The largest gains are observed for QA and NER with larger multilingual models such as XLM-R and mBERT, while smaller monolingual models (BanglaBERT, BanglighBERT) show moderate improvements. For example, Sentiment Analysis throughput increased from 22.2 to 36.6 samples/sec for BanglaBERT (65%) and from 13.99 to 21.38 samples/sec for XLM-R (82%). These results demonstrate that AMP not only reduces memory usage but also maximizes hardware utilization, enabling faster model training across diverse Bangla NLP tasks.

**Training Time.** AMP consistently reduces the average time per epoch across all models and tasks. Task-wise reductions



**Fig. 2:** Comparison of FP32 and AMP training efficiency across four Bangla NLP tasks.

range from 34–64%, with the largest gains observed in QA and NER. For instance, Sentiment Analysis epoch time decreased from 530.7s to 321.5s for BanglaBERT (39.4%) and from 840.0s to 549.9s for XLM-R (45.2%). Similarly, QA training time dropped by 46–64% depending on the model. Smaller monolingual models like BanglighBERT benefit moderately, while larger multilingual models such as XLM-R and mBERT achieve the highest speedups. These reductions, combined with memory savings and increased throughput, demonstrate AMP’s effectiveness in accelerating Bangla NLP training without compromising model performance.

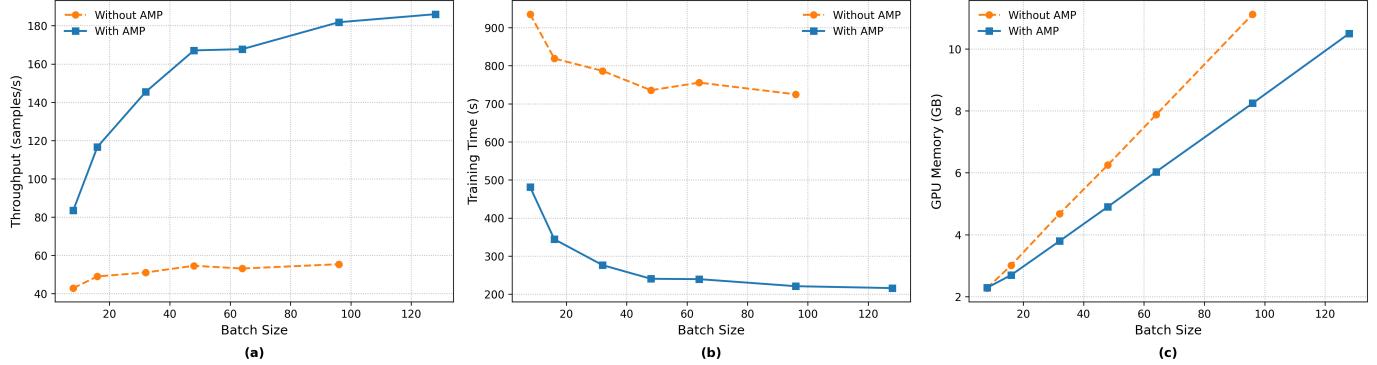
Overall, AMP delivers substantial efficiency with lower memory consumption, higher throughput, and reduced training time, while preserving model performance. These results highlight the practicality of AMP for Bangla NLP research in hardware-limited settings.

#### C. Impact of Batch Size on Training Performance

To further explore practical strategies for improving training efficiency, we analyze how varying batch sizes influ-

ence performance and resource utilization under FP32 and AMP regimes. We investigate the effect of batch size on training efficiency for the Error Classification task using the BanglaBERT model. Experiments were conducted under both FP32 and Automatic Mixed Precision (AMP), measuring per-epoch training time, throughput, and peak GPU memory across batch sizes of 8, 16, 32, 48, 64, 96, and 128. Figure 3 shows that AMP enables larger batch sizes without exceeding GPU memory limits. FP32 training failed at batch size 128 due to out-of-memory errors, whereas AMP successfully processed this batch with peak memory usage of 10.5 GB. Across the batch size range, AMP consistently improved efficiency: throughput increased by 95–228%, per-epoch training time decreased by 49–70%, and peak memory usage was reduced by up to 26%.

These findings highlight AMP’s ability to accelerate training while enabling larger batch sizes, improving GPU utilization, and maintaining model performance in resource-constrained environments.



**Fig. 3:** Comparison of AMP and full-precision training across batch sizes. (a) Throughput, (b) Training time, (c) GPU memory usage. AMP substantially improves throughput and reduces training time without significant memory overhead.

#### D. Discussion

Our results demonstrate that Automatic Mixed Precision (AMP) substantially improves computational efficiency for Bangla NLP tasks without compromising performance. Peak GPU memory usage decreased by up to 42%, training throughput increased by as much as 179%, and per-epoch training time dropped by up to 64%. Task metrics, including Accuracy, F1, and Exact Match, remain largely unchanged, with differences within 1–2 percentage points. AMP also enables larger batch sizes, improving GPU utilization and convergence. These findings establish AMP as a practical, efficient approach for training transformer-based Bangla NLP models in resource-constrained environments.

#### V. CONCLUSION

This research systematically compared full-precision (FP32) and automatic mixed precision (AMP) training across multiple Bangla NLP tasks, including sentiment analysis, named entity recognition, error classification, and question answering. Experimental results demonstrate that AMP substantially reduces peak GPU memory consumption, decreases per-epoch training time, and increases throughput, all without compromising task performance. These findings are particularly valuable for Bangla, where computational resources are often limited, enabling more efficient model development and broader experimentation. Future work includes scaling AMP to larger and multilingual pretrained models, integrating complementary techniques such as gradient checkpointing and quantization, evaluating performance on domain-specific, dialectal, and code-mixed corpora, and exploring energy-efficient training and real-time inference on resource-constrained hardware to advance sustainable and practical Bangla NLP applications.

#### REFERENCES

- [1] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: state of the art, current trends and challenges,” *Multimedia Tools and Applications*, vol. 82, pp. 3713 – 3744, 2017.
- [2] P. Kumar, “Large language models (llms): survey, technical frameworks, and future challenges,” *Artif. Intell. Rev.*, vol. 57, p. 260, 2024.
- [3] A. Bhattacharjee, T. Hasan, K. S. Mubasshir, M. S. Islam, W. U. Ahmad, A. Iqbal, M. S. Rahman, and R. Shahriyar, “Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla,” in *NAACL-HLT*, 2021.
- [4] A. Bhattacharjee, T. Hasan, W. U. Ahmad, and R. Shahriyar, “Banglanlg and banglat5: Benchmarks and resources for evaluating low-resource natural language generation in bangla,” in *Findings*, 2022.
- [5] P. Bhattacharyya and A. Bhattacharya, “Banglabyt5: Byte-level modelling for bangla,” *ArXiv*, vol. abs/2505.17102, 2025.
- [6] M. S. Salim, H. Murad, D. Das, and F. Ahmed, “Banglapgpt: A generative pretrained transformer-based model for bangla language,” *2023 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*, pp. 56–59, 2023.
- [7] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, “Mixed precision training,” *ArXiv*, vol. abs/1710.03740, 2017.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *North American Chapter of the Association for Computational Linguistics, 2019*.
- [9] S. Khanuja, D. Bansal, S. Mehtani, S. Khosla, A. Dey, B. Gopalan, D. K. Margam, P. Aggarwal, R. T. Nagipogu, S. Dave, S. Gupta, S. C. B. Gali, V. Subramanian, and P. Talukdar, “MuriL: Multilingual representations for indian languages,” 2021.
- [10] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” *ArXiv*, vol. abs/1911.02116, 2019.
- [11] A. K. Zehady, S. A. Mamun, N. Islam, and S. Karmaker, “Bongllama: Llama for bangla language,” *ArXiv*, vol. abs/2410.21200, 2024.
- [12] N. Raihan and M. Zampieri, “Tigerllm - a family of bangla large language models,” *ArXiv*, vol. abs/2503.10995, 2025.
- [13] O. Kuchaiev, B. Ginsburg, I. Gitman, V. Lavrukhin, J. Li, H. Nguyen, C. Case, and P. Micikevicius, “Mixed-precision training for nlp and speech recognition with openseq2seq,” *arXiv: Computation and Language*, 2018.
- [14] C. Zhao, T. Hua, Y. Shen, Q. Lou, and H. Jin, “Automatic mixed-precision quantization search of bert,” in *International Joint Conference on Artificial Intelligence*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:237100483>
- [15] M. A. Hasan, F. Alam, A. Anjum, S. Das, and A. Anjum, “Blp-2023 task 2: Sentiment analysis,” *ArXiv*, vol. abs/2310.16183, 2023.
- [16] M. A. Hasan, S. Das, A. Anjum, F. Alam, A. Anjum, A. Sarker, and S. R. H. Noori, “Zero- and few-shot prompting with llms: A comparative study with fine-tuned models for bangla sentiment analysis,” *ArXiv*, vol. abs/2308.10783, 2023.
- [17] K. I. Islam, S. Kar, M. S. Islam, and M. R. Amin, “Sentnob: A dataset for analysing sentiment on noisy bangla texts,” in *Conference on Empirical Methods in Natural Language Processing*, 2021.
- [18] A. Khan, F. Kamal, N. Nowher, T. Ahmed, S. Ahmed, and T. M. Chowdhury, “Nervous about my health: Constructing a bengali medical named entity recognition dataset,” in *Conference on Empirical Methods in Natural Language Processing*, 2023.

- [19] N. T. Oshin, S. Hoque, M. Fahim, A. A. Ali, M. A. Amin, and A. M. Rahman, “Bateclacor: A novel dataset for bangla text error classification and correction,” *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, 2023.
- [20] S. M. S. Ekram, A. A. Rahman, M. S. Altaf, M. S. Islam, M. M. Rahman, M. M. Rahman, M. A. Hossain, and A. R. M. Kamal, “Banglarqa: A benchmark dataset for under-resourced bangla language reading comprehension-based question answering with diverse question-answer types,” in *Conference on Empirical Methods in Natural Language Processing*, 2022.