Independent University **Bangladesh (IUB)**

IUB Academic Repository

Graduate Thesis 2026

2026-04

DrishtiXAI: a Bengali image captioning system with explainable AI

Sakib, Abu Musa

Independent University, Bangladesh (IUB)

https://ar.iub.edu.bd/handle/11348/1598

Downloaded from IUB Academic Repository



DrishtiXAI: A Bengali Image Captioning System with Explainable AI

By

Abu Musa Sakib
ID: 2321673

Sumyia Afnan Mukta
ID: 2321672

Supervisor:
Ashraful Islam, PhD
Assistant Professor
Department of Computer Science and Engineering

Co-Supervisor:
Rashedur Rahman, D.Eng.
Assistant Professor
Department of Computer Science and Engineering

April, 2026

Thesis submitted in partial fulfillment for the degree of Master of Science in Computer Science

Department of Computer Science and Engineering
Independent University, Bangladesh

Attestation

We are aware that plagiarism is against the rules and regulations of our university and is completely forbidden. We guarantee that our work is authentic. In our project, we have used some copyrighted material and models of others which has been properly cited by us, using the international standards followed by proper guidelines.

Author Name:

. . . . . .

Signature:

. . . . . .

Author Name:

. . . . . .

Signature:

. . . . . .

Evaluation Committee

Supervisor

Name: _____ Signature: _____

Internal Examiner 1

Name: _____ Signature: _____

Internal Examiner 2

Name: _____ Signature: _____

External Examiner

Name: _____ Signature: _____

Acknowledgement

We are deeply grateful to our respected supervisor, Ashraful Islam, PhD, Director, Center for Computational & Data Sciences (CCDS) and Assistant Professor, Department of Computer Science and Engineering, Independent University, Bangladesh (IUB). The success of our work was greatly aided by his committed leadership, perceptive advice, extraordinary patience, insightful suggestions and the kind donation to his valuable time. We sincerely appreciate his expertise and direction, which have greatly enhanced the caliber of our academic journey and the breadth of our work.

Additionally, we would like to express our sincere gratitude to our co-supervisor, Md. Rashedur Rahman, D. Eng., Assistant Professor, Department of Computer Science and Engineering, Independent University, Bangladesh (IUB). His technical knowledge, unwavering support, and insightful recommendations have significantly contributed to the project's success. We value the time he spent reviewing our work and offering insightful criticism that helped refine our research.

We would like to thank the Department of Computer Science and Engineering and the Center for Computational & Data Sciences (CCDS) for their excellent support throughout this project. The collaborative and creative environment at the CSE department and the CCDS Lab helped us develop and implement our ideas more effectively. Their encouragement has been important in shaping our research direction. We appreciate the resources and opportunities they provided, which improved our learning experience.

We also thank everyone who contributed to our project work, whether directly or indirectly.

Abstract

Automated image captioning connects computer vision and Natural Language Processing (NLP) to create human-like descriptions of images. Deep learning has changed this field, but most research focuses on English, leaving languages like Bengali with far less attention. Standard models also work as "black boxes," which creates problems for applications that need transparency, like assistive technology. In this work, we present **DrishtiXAI**, an explainable Bengali image captioning system built to help visually impaired users. Since culturally relevant resources for Bengali are scarce, we created a dataset of **18,539 Bengali image-caption pairs**, organized into categories like humans, animals, vehicles, and natural scenes to cover common visual contexts in Bangladesh. We tested two different approaches to find the best balance between performance and efficiency: a classic **InceptionV3 encoder with Bahdanau attention and Gated Recurrent Unit (GRU) decoder** ("Show, Attend, and Tell") versus a modern **EfficientNet-B4 backbone with a Transformer decoder**. Our results showed that while the Transformer model trained quickly, it overfit badly (validation loss climbed from 22.2). The InceptionV3-GRU model performed better, staying stable (validation loss around 1.05) and proving more reliable. Testing on a separate dataset gave us a **mean BLEU score of 0.5157** and a **median BLEU of 0.4609**. We also checked performance using **ROUGE-1**, **Token Jaccard Similarity**, and **Character-Level Levenshtein Ratio** metrics, which confirmed the model generates fluent and appropriate Bengali captions. Beyond numbers, we focused on interpretability using **Bahdanau attention mechanisms**. These create spatial attention maps that show which parts of an image the model looks at when generating each word (for example, highlighting a rickshaw when producing the word "vehicle"). Recognizing that traditional n-gram metrics fail to capture the nuances of transparency and trust, we justify and implement a multi-dimensional evaluation framework that combines visual grounding analysis with human-centric usability metrics. We built this model into a mobile app with on-device Text-to-Speech (TTS) and visual attention displays. A user study with 20 participants showed the system works well in practice, with high ratings for **Clarity (5.80/7)** and **Supportiveness (5.65/7)**. This research shows the important trade-offs between model complexity and limited data in low-resource languages, and proves that transparent, accessible, and culturally appropriate AI for Bengali speakers is possible.

List of Abbreviations

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
ROI	Region of Interest
LCS	Longest Common Subsequence
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
BP	Brevity Penalty
SCCE	Sparse Categorical Cross-Entropy
NLP	Natural Language Processing
TTS	Text-to-Speech
API	Application Programming Interface

Contents

List of Abbreviations **5**

1 Introduction **11**
1.1 Background . . . 11
1.2 Problem Statement . . . 11
1.3 DrishtiXAI: Project Outcomes and Contributions . . . 12
1.4 Rationale for Multi-Dimensional Evaluation . . . 12
1.5 Research Questions . . . 12
1.6 Report Outline . . . 13

2 Literature Review **15**
2.1 Scope and Objectives . . . 15
2.2 Systematic Review Methodology . . . 15
2.2.1 Search Strategy . . . 15
2.2.2 Document and Subject Filters . . . 16
2.2.3 Deduplication and Reference Management . . . 16
2.2.4 Inclusion and Exclusion Criteria . . . 16
2.2.5 Screening . . . 17
2.2.6 Study Selection Process via PRISMA . . . 17
2.2.7 Data Extraction and Synthesis . . . 19
2.3 Bengali Image Captioning: State of the Art . . . 19
2.3.1 Key Studies and Contributions . . . 19
2.3.2 Architectural Trends and Gaps . . . 19
2.4 Explainable AI in Image Captioning . . . 20
2.4.1 Attention Mechanisms as Interpretability Tools . . . 20
2.4.2 Attribution Methods and Saliency Maps . . . 21
2.4.3 Region-Word Alignment and Cross-Modal Grounding . . . 21
2.5 Identified Research Gaps . . . 22
2.6 Bibliometric Findings . . . 22
2.7 Recommendations for Future SLRs and Studies . . . 22

3 Methodology **23**
3.1 Motivation and Purpose . . . 23
3.2 Dataset Sources . . . 23
3.2.1 Training Dataset Composition . . . 23
3.2.2 Testing Dataset Composition . . . 24
3.2.3 Caption Annotation and Quality Assurance . . . 26
3.2.4 Vocabulary and Linguistic Statistics . . . 27
3.3 Preparing the Dataset for Model Training . . . 29
3.3.1 Category Rationales and Query Choices . . . 30

3.4 Dataset Structure and Core Features . . . 31
3.4.1 Diversity of Visual Categories . . . 31
3.4.2 Caption Characteristics . . . 32
3.5 Pipeline and Operational Details . . . 32
3.5.1 Data and Model Training . . . 32
3.5.2 Pipeline Deployment . . . 32
3.5.3 Quality Check . . . 32
3.5.4 Experimental Setup . . . 34
3.5.5 Real World Deployment . . . 34
3.5.6 Operational Considerations . . . 35
3.6 Model . . . 36
3.6.1 Model Description . . . 36
3.6.2 Model Architecture of InceptionV3 . . . 37
3.6.3 Model Training . . . 41
3.6.4 Training Algorithm . . . 42
3.6.5 Model Architecture of EfficientNetB4 . . . 42
3.6.6 Evaluation Metrics . . . 45
3.7 Mobile Application . . . 48
3.7.1 Primary Features . . . 48
3.7.2 How Captioning and Attention Maps Work . . . 49
3.7.3 Color Palette and Background Processing . . . 49
3.7.4 Integration and Deployment . . . 49

4 Results and Analysis **51**
4.1 Model Outputs . . . 51
4.1.1 Performance Breakdown . . . 51
4.1.2 Understanding “Attention” . . . 52
4.1.3 Common Challenges . . . 52
4.2 Training Dynamics and Loss Behavior . . . 56
4.2.1 Training Loss Progression . . . 56
4.2.2 Validation Loss Trajectory and Overfitting . . . 57
4.2.3 Early Stopping Mechanism and Model Selection . . . 57
4.2.4 Analysis: Convergence State and Early Stopping Sufficiency . . . 58
4.2.5 Validation Metrics and Caption Quality . . . 59
4.2.6 Implications for Model Deployment . . . 60
4.3 Model Evaluation Results . . . 60
4.3.1 Evaluation Metrics Across Training Epochs . . . 61
4.3.2 Per-Image Metric Distributions . . . 62
4.3.3 ROUGE Score Distributions . . . 64
4.3.4 Quantitative Performance Summary . . . 65
4.3.5 Qualitative Analysis and Error Patterns . . . 66
4.3.6 Key Findings and Implications . . . 67
4.4 Evaluation Survey . . . 68
4.4.1 Summary Statistics . . . 68
4.4.2 Usability Instrument . . . 69
4.4.3 Plots and Visual Analysis . . . 69
4.4.4 Final Interpretation of User Feedback . . . 74
4.4.5 Implications and Next Steps . . . 75
4.5 Evaluating Explainable AI: A Multi-Dimensional Rationale . . . 76

5 Model Comparison: EfficientNetB4 vs. InceptionV3 **77**
5.1 Overview . . . 77
5.1.1 EfficientNet: Precision through Compound Scaling . . . 77
5.1.2 InceptionV3: The Visual Encoder . . . 80
5.1.3 Justification for Architecture Selection . . . 82
5.2 Qualitative Outputs and Attention . . . 83
5.3 Comparative Analysis and Caveats . . . 84
5.3.1 Convergence . . . 84
5.3.2 Stability . . . 84
5.3.3 Interpretability . . . 84
5.3.4 Decoding Behavior and Its Effects . . . 85
5.3.5 Limitations of Comparison . . . 86
5.4 Summary . . . 86

6 Discussion **87**
6.1 Challenges . . . 87
6.2 Threats to Validity . . . 88
6.3 Research Questions and Answers . . . 89

7 Conclusion and Future Work **90**
7.1 Conclusion . . . 90
7.2 Future Work . . . 90
7.2.1 Advancing Model Architectures . . . 90
7.2.2 Improving Attention, Alignment, and Localization . . . 91
7.2.3 Dataset Expansion and Annotation Improvements . . . 91
7.2.4 Leveraging Pretraining and Transfer Learning . . . 91
7.2.5 Improving Training Efficiency . . . 91
7.2.6 Mitigating Dataset and Model Bias . . . 91
7.2.7 Enhancing Caption Fluency . . . 91
7.2.8 Developing APIs for Organizational Integration . . . 91
7.2.9 Quantitative Metrics for XAI Faithfulness . . . 92

Bibliography **92**

List of Figures

1 PRISMA Flow Diagram for Systematic Literature Review 18

2 Training and Testing Dataset Composition 26
3 Model Training, Evaluation, and Deployment Workflow Pipeline 33
4 Orange Pi Zero3 (Image Source: [1]) 34
5 Encoder–Decoder Architecture with Bahdanau Attention for Bengali Image Captioning 38
6 Flowchart of the Bahdanau Attention Mechanism. 40
7 EfficientNetB4 backbone and projection head for computing image embeddings. 44
8 User interface of different features available in interactive mobile application of DrishtiXAI app 50

9 Qualitative model outputs: representative high-performing and typical examples. Each panel shows the input image, reference and generated Bengali captions, and BLEU score; attention overlays are shown where available. . 53
10 Typical-performing examples illustrating common success and failure modes; each panel lists the reference and generated Bengali captions and BLEU score. 54
11 Worst-performing examples illustrating failure modes; each panel lists the reference and generated Bengali captions and BLEU score. 55
12 Training and validation loss across 23 epochs for the Bengali image captioning model. The green dashed line at epoch 3 indicates the early stopping trigger point, while the shaded region shows training that proceeded beyond the optimal checkpoint. Final model weights were restored from epoch 3, corresponding to the lowest validation loss. 56
13 Evolution of evaluation metrics across 23 training epochs. The plot tracks six metrics: precision (blue), recall (orange), F1 score (green), normalized exact match (red), token Jaccard similarity (purple), and character-level ratio (brown). The annotation marks the best F1 score of 0.326 at epoch 2, showing the optimal performance point before overfitting reduced caption quality. 61
14 Per-image distribution analysis of caption quality metrics. Top row: F1 score distribution (left) and Precision distribution (right) across test set images. Bottom row: Recall distribution (left) and correlation between F1 score and token Jaccard similarity (right). The histograms show substantial variation in model performance, with F1 scores concentrated in the 0.1–0.4 range, while the scatter plot shows strong positive correlation ($r \approx 0.85$) between F1 and token Jaccard metrics. 62

15 Distribution of ROUGE F1 scores across test set images. ROUGE-1 measures unigram overlap, ROUGE-2 captures bigram matching, and ROUGE-L evaluates Longest Common Subsequence (LCS) alignment. The progressively left-skewed distributions from ROUGE-1 to ROUGE-2 reflect increasing difficulty in achieving higher-order n-gram matches, with ROUGE-2 showing pronounced concentration near zero. 64
16 Distribution of semantic-item responses (boxplots, $n = 20$). Median scores are consistently 6 for most dimensions (Supportive, Complexity, Efficiency, Clarity, Engagement, Interest, Inventiveness), with IQRs indicating moderate to strong consensus (Q1 $\approx$ 4–5, Q3 = 7). The Technology dimension shows lower ratings (median = 4, mean = 3.85), while the Overall rating demonstrates strong satisfaction (median = 6, mean = 5.95). 71
17 Distribution of SUS-like usability scores (0–100) for DrishtiXAI. The histogram shows that most participant ratings cluster between 45 and 60, indicating generally acceptable to good perceived usability, with limited dispersion and a small number of lower and higher outliers. 71
18 Mean scores for usability questionnaire items. Strengths: Integration (4.47), Confidence (4.29), Ease of use (4.12). Weaknesses: Consistency (2.24), Cumbersomeness (2.24), Complexity (3.12). 72
19 Stacked Likert summary of usability responses ($n = 20$). The chart displays proportional response distributions across a 5-point scale (5 = Strongly Agree to 1 = Strongly Disagree). Strongest agreement observed for Q4 (Integration) and Q7 (Confidence); most polarized/negative responses for Q3 (Consistency) and Q6 (Cumbersome). Color-coded segments indicate the proportion of respondents in each rating category per item. 73

20 EfficientNetB4 Model Output over a set of images 79

1 Introduction

1.1 Background

Image captioning combines computer vision and NLP to generate human-like descriptions of images. Modern systems use an encoder-decoder approach: a Convolutional Neural Network (CNN) extracts visual features from the image, and a Recurrent Neural Network (RNN) or Transformer generates a caption based on those features [2, 3]. Attention mechanisms improve this process by allowing the model to focus on relevant parts of the image while generating each word [4].

Most image captioning research focuses on English and uses datasets like MS COCO, Flickr8K, and Flickr30K [5, 6]. This creates a problem for other languages, especially those with different grammar structures and cultural contexts. Bengali, one of the world's most spoken languages, has very limited resources for image captioning research [7]. While projects like Chittron and TextMage have shown that Bengali captioning is possible, they face challenges with small datasets, limited caption variety, and poor generalization [6, 5].

Another important challenge is making these systems explainable. Deep learning models are often "black boxes"; it is hard to understand why they generate specific words. Explainable AI (XAI) techniques like attention visualization, Grad-CAM, and LIME help reveal how models make decisions [8, 9, 10]. This is especially important for assistive technologies, where users need to trust the system. For visually impaired users, combining image captions with TTS and visual explanations can significantly improve accessibility [11, 12, 13, 14, 15].

1.2 Problem Statement

Despite progress in image captioning, creating accurate and useful captions in Bengali faces three main challenges:

First, Bengali has limited training data. Most captioning models are trained on large English datasets and do not work well for Bengali because of differences in grammar, word structure, and meaning [16]. Existing Bengali datasets are small, noisy, and lack diverse captions, which causes models to overfit and perform poorly on new images [13].

Second, current systems lack transparency. When a model generates an incorrect caption, it is difficult to understand why or fix the problem. This is particularly problematic for assistive applications where users depend on accurate descriptions.

Third, deploying these systems on mobile devices is challenging. Models must be small enough to run on phones, fast enough for real-time use, and reliable enough for practical applications [17].

1.3 DrishtiXAI: Project Outcomes and Contributions

This project presents **DrishtiXAI**, an explainable Bengali image captioning system designed for accessibility and practical use. The system makes the following contributions:

- **Bengali-Adapted Architecture:** We developed an encoder-decoder pipeline specifically designed for Bengali, handling its complex grammar, rich morphology, and limited vocabulary. The system uses attention mechanisms to improve alignment between images and captions, addressing common problems like repetitive phrases and missing objects [14, 15].
- **Explainable AI Integration:** The system includes attention visualization that shows which parts of an image the model focuses on when generating each word. This makes the model's decisions transparent and helps identify errors [11, 12, 18].
- **Text-to-Speech for Accessibility:** We integrated on-device Bengali TTS, allowing visually impaired users to hear image descriptions. This makes visual content accessible through audio.

1.4 Rationale for Multi-Dimensional Evaluation

Evaluating an explainable AI system poses unique challenges that traditional metrics alone cannot address. While BLEU and ROUGE provide a quantitative measure of linguistic similarity, they fail to capture whether a model's underlying reasoning is faithful to the image or if it fosters genuine user trust. Given that **DrishtiXAI** is targeted at assistive technology, we justify a multi-dimensional evaluation framework. This includes: (1) *quantitative n-gram metrics* for linguistic quality, (2) *qualitative attention analysis* to verify visual grounding and faithfulness, and (3) *human-centric usability surveys* to assess practical trust and clarity. This comprehensive approach ensures that our system is evaluated not just on its ability to "speak" Bengali, but on its capacity to explain its decisions transparently to the user.

1.5 Research Questions

This project is guided by the following research questions, which frame the empirical work and the literature review:

1. **Can attention-based models generate accurate Bengali captions with limited training data?**

   This question is motivated by the dataset scarcity and morphological challenges described above and is evaluated through our experiments reported in the Results and Analysis chapter.

   Discussed in Chapter 4: Results and Analysis, which presents the quantitative evaluation metrics and performance analysis.

2. **Do attention visualizations improve user trust and help diagnose errors in Bengali captioning systems?**

   We investigate attention maps and attribution techniques alongside user-oriented interpretability discussions in Related Work and Discussion to assess practical trust gains.

   Discussed in Chapter 4: Results and Analysis (Qualitative Analysis) and Chapter 6: Discussion, covering the user evaluation survey and trust assessment.

3. **How do different CNN encoders (EfficientNet-B4 vs. InceptionV3) compare in terms of caption quality, generalization, and efficiency for Bengali?**

   This RQ connects directly to our comparative encoder experiments (EfficientNet-B4 vs. InceptionV3) and to deployment considerations for resource-constrained devices.

   Discussed in Chapter 5: Model Comparison: EfficientNetB4 vs. InceptionV3, which details the architectural trade-offs and training dynamics of both models.

4. **Which dataset strategies (augmentation, multiple captions per image, vocabulary design) work best for Bengali's complex grammar?**

   The question ties into the Dataset and Methodology chapters and the SLR recommendations favoring multi-caption annotation, morphology-aware tokenization, and balanced sampling.

   Discussed in Chapter 3: Methodology (Dataset Preparation) and Chapter 6: Discussion, analyzing the impact of data curation quality.

5. **Can TTS be integrated on mobile devices without sacrificing speed or reliability**?

   This addresses system-level design and deployment trade-offs discussed in the Outcome and Discussion sections, where latency, model compression, and versioning are evaluated.

   Discussed in Chapter 6: Discussion, highlighting system feasibility and user feedback.

1.6 Report Outline

The remainder of this report is organized as follows:

- **Chapter 2: Literature Review** systematically reviews prior research on automatic image captioning, explainable vision–language models, and Bengali captioning. It identifies dominant paradigms and unresolved challenges that motivate our work.
- **Chapter 3: Methodology** details our dataset construction process and the proposed model architectures. It describes the data collection sources, preprocessing steps, and the implementation of our primary InceptionV3-based encoder-decoder framework.

- **Chapter 4: Results and Analysis** presents the experimental evaluation of our system. It includes quantitative performance metrics (BLEU scores), qualitative analysis of model outputs, and findings from our user evaluation survey.
- **Chapter 5: Model Comparison: EfficientNetB4 vs. InceptionV3** provides a comparative analysis of two different encoder architectures. It evaluates their training dynamics and justifies the selection of our final model based on performance and stability.
- **Chapter 6: Discussion** synthesizes our findings, analyzing the trade-offs between architectural choices, the impact of data scarcity, and the broader implications of deploying explainable AI for accessibility.
- **Chapter 7: Conclusion and Future Work** summarizes the project's contributions and limitations. It outlines potential future directions, including dataset expansion and advanced architectural improvements.

2 Literature Review

This chapter systematically reviews prior research on automatic image captioning, explainable vision–language models, and Bengali or low-resource language captioning. The review identifies dominant architectural paradigms, datasets, and evaluation practices, and highlights unresolved challenges that motivate the DrishtiXAI framework.

2.1 Scope and Objectives

The SLR aims to answer: which methods, datasets, and evaluation practices have been used for explainable image captioning, how these approaches generalize to low-resource languages (especially Bengali), and what gaps remain for building deployable, accessible systems. We prioritized peer-reviewed articles, major conference proceedings, and high-quality preprints published between 2018 and 2025.

2.2 Systematic Review Methodology

The PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines were followed when conducting a systematic literature review (SLR) in order to guarantee accuracy in methodology, transparency, and reproducibility. Peer-reviewed journal articles, conference proceedings, and a few high-impact preprints published between 2018 and 2025 that are related to explainable AI (XAI), image captioning, and Bengali or other low-resource languages are the main focus of the review.

2.2.1 Search Strategy

The **Scopus** database served as the main indexing source for the primary literature search, which was supplemented by IEEE Xplore, the ACM Digital Library, SpringerLink, and arXiv to guarantee thorough coverage. Both language-specific and general explainability literature were captured by applying exact Boolean queries to titles, abstracts, and keywords. Two primary query formulations were used in the search strategy:

Primary Search Query (SQ1):

```
( ( "image captioning" OR "image description" OR "image-to-text" )
AND ( bangla OR bengali ) AND ( explain* OR xai OR interpret* ) )
```

Secondary Search Query (SQ2):

```
( ( "image captioning" OR "image description" OR "image-to-text" )
AND ( explain* OR xai OR interpret* ) )
```

Temporal filters (2018–2025) were applied for trend analyses; broader queries without language filters were used to capture advances in explainability applicable to low-resource settings.

Table 2.1: Temporal Distribution of Explainable Image Captioning Research (Scopus)

Year	**Publications**	**Year**	**Publications**
2018	22	2022	61
2019	35	2023	98
2020	38	2024	138
2021	45	2025	147
Total (2018–2025)		**584**	

The temporal distribution of publications found in the Scopus search is summarized in Table 2.1. According to the data, research output increased exponentially between 2018 and 2025, with publications nearly increasing by three times. Demands for transparency, reliability, and regulatory compliance in AI systems are driving this trend, which highlights the growing significance of explainability in vision-language models.

We searched for titles, abstracts, and keywords on the Scopus database using the formulated search queries. 722 documents were returned by the secondary search query (SQ2) without date restrictions, indicating the significant increase in research interest in explainable image captioning, especially in recent years. According to Table 2.1, an analysis of publication trends showed that 80.9% of all identified research (584 papers) was published between 2018 and 2025, with a significant acceleration after 2020. In particular, the number of papers published annually rose from 22 in 2018 to 147 in 2025, indicating the increasing convergence of explainable AI, computer vision, and natural language processing.

To find studies that particularly addressed Bengali image captioning and low-resource language scenarios, more focused searches were carried out using the primary search query (SQ1). Four directly relevant studies on the creation of image descriptions in Bengali were found through these complementary searches.

2.2.2 Document and Subject Filters

To preserve methodological quality we limited results to the following document types: journal articles, conference papers, and review articles. Subject area filters focused on Computer Science, Artificial Intelligence, and Computational Linguistics to avoid scope drift.

2.2.3 Deduplication and Reference Management

Search results from multiple sources were exported to a reference manager and deduplicated before screening. Duplicate detection prioritized DOI matching and, where DOI was absent, title-author-year combinations. Deduplication and export details (software and timestamps) are recorded in the supplementary materials for reproducibility.

2.2.4 Inclusion and Exclusion Criteria

Studies were included if they satisfied at least one of the following conditions: (i) proposed or evaluated image captioning models with explainability mechanisms; (ii) addressed Ben-

gali or other low-resource languages in vision–language tasks; or (iii) incorporated attention visualization, attribution methods, or vision–language alignment techniques with interpretability focus.

Studies were excluded if they focused exclusively on unrelated computer vision or NLP tasks, provided no empirical evaluation, addressed only English captioning without transferable methodological contributions, or consisted of non-peer-reviewed content such as editorials, tutorials, or blog posts.

Table 2.2: Inclusion and Exclusion Criteria for the Systematic Literature Review

Criterion	**Inclusion**	**Exclusion**
Data Sources	Scopus, IEEE Xplore, ACM DL, SpringerLink, arXiv	Blogs, non-indexed repositories, unpublished manuscripts
Publication Type	Journal articles, conference papers, high-impact preprints	Editorials, surveys without experiments, tutorials
Research Focus	Image captioning, vision–language models, XAI, low-resource languages	Unrelated CV or NLP tasks
Language Scope	Bengali-focused or transferable multilingual studies	English-only studies with no methodological transfer
Explainability	Attention, attribution, region-word alignment, XAI	Black-box models without interpretability
Time Period	2018–2025	Published before 2018

2.2.5 Screening

We applied a two-stage screening process (title/abstract screening followed by full-text review). Where available, dual independent screening was used and inter-rater disagreements were resolved by consensus.

2.2.6 Study Selection Process via PRISMA

The study selection followed a four-stage PRISMA process: identification, screening, eligibility assessment, and inclusion. The initial Scopus search identified 142 records, supplemented by 18 additional records through reference screening and forward citation tracking, yielding a total of 160 records. After removing 32 duplicate entries, 128 unique records underwent title and abstract screening. Of these, 82 records were excluded based on irrelevance to the research scope (e.g., studies focusing solely on object detection, sentiment analysis, or text-only NLP without vision components).

The full-text retrieval of the remaining 46 studies was the goal, and every article was successfully retrieved. 31 more studies were excluded from the full-text eligibility assessment because they lacked explainability mechanisms, had insufficient technical detail, or lacked empirical evaluation. Fifteen studies that satisfied all inclusion criteria and made significant contributions to explainable image captioning, low-resource language captioning, or both made up the final accumulation.

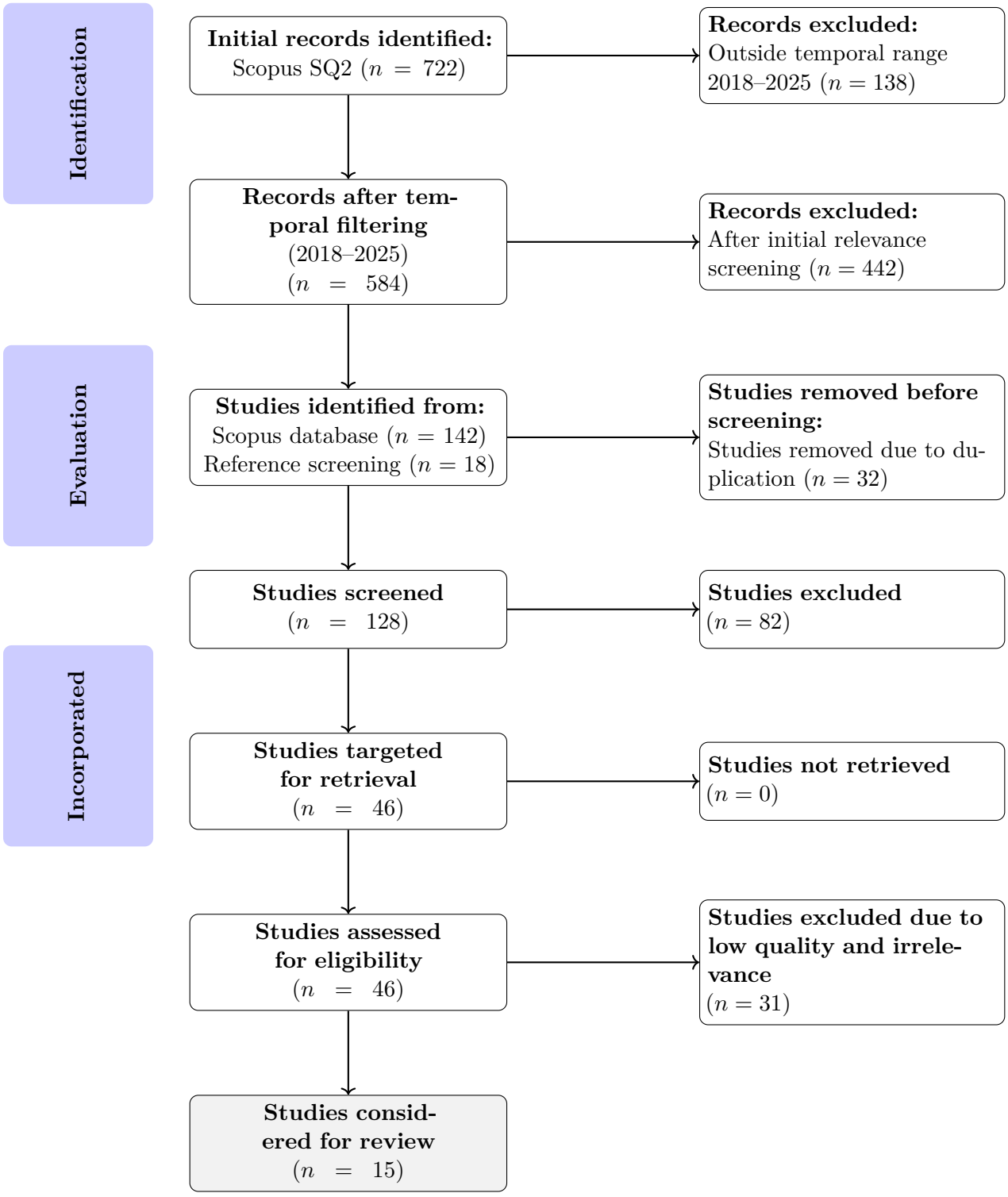


Figure 1: PRISMA Flow Diagram for Systematic Literature Review

2.2.7 Data Extraction and Synthesis

For each included study, structured data were extracted covering: dataset characteristics (size, language, domain), encoder–decoder architecture (CNN type, RNN/Transformer variant), attention or explainability mechanism (visual attention, cross-modal alignment, attribution methods), evaluation metrics (BLEU-1 through BLEU-4, ROUGE-L), and deployment considerations (computational efficiency, real-time capability).

We narratively synthesized the evidence to pinpoint major trends, architectural breakthroughs, evaluation methods, and current limitations. We paid special attention to research on Bengali and other low-resource languages, as these are major gaps in the field. Out of the 15 studies we reviewed, only four focused on Bengali image captioning. This really highlights how few resources and models exist for the language, even though it is the fifth most spoken in the world with over 250 million speakers.

2.3 Bengali Image Captioning: State of the Art

We do not have much research on Bengali image captioning compared to English and other well-resourced languages. The main problems are limited annotated datasets, few pre-trained language models, and the complex structure of Bengali script. But recent studies have started to address these issues.

2.3.1 Key Studies and Contributions

Das et al. (2023) [19] used a visual attention model to caption Bengali images. By using an encoder-decoder setup, they generated sentences that were both grammatically sound and meaningful. They also cleaned up existing datasets to fix labeling errors, ultimately reaching BLEU-1 scores of 0.67 and 0.65 across two different Bengali datasets. While they proved that models built for English can work for Bengali, a large performance gap still exists between the two languages.

Faruk et al. (2020) [20] created the BNATURE dataset, which includes 8,000 images paired with five Bengali captions each. Their system used InceptionV3 to process images and a Bidirectional Gated Recurrent Unit (BiGRU) to handle the text, testing both argmax and beam search methods. They reported scores of 42.6 for BLEU-1 and 28.7 for METEOR. This was a major step in building Bengali datasets, even though the scale is still much smaller than English standards like MS-COCO.

Jishan et al. (2020) [21] introduced the Bangla Natural Language Image to Text (BNLIT) dataset. They built a model that paired Convolutional Neural Networks (CNN) with Bidirectional Long Short-Term Memory (BiLSTM) networks. Their goal was to keep the model accurate without making it too computationally heavy, focusing specifically on how contextual and semantic details improve caption quality.

Together, these studies set the stage for Bengali image captioning but highlight a few clear problems: datasets are still too small for large-scale training, there is little focus on how these models actually reach their conclusions, and testing relies mostly on numbers rather than human checks for cultural or linguistic nuances. Additionally, it is still unclear how well these models perform when applied to entirely new datasets.

2.3.2 Architectural Trends and Gaps

Most Bengali captioning systems use the standard encoder-decoder setup with attention mechanisms, similar to the models used for English. However, several problems still make

these systems less effective:

- **Poor Explainability:** While attention weights give a rough idea of how a model works, current systems do not show exactly how specific words align with regions in an image. We lack clear visualizations or explanations that would help someone understand why a model chose a particular caption over another.
- **Language Complexity:** Bengali is linguistically complex, featuring diacritics, compound characters, and various vowel signs. The way current models break down and represent text often misses these details, which lowers the overall quality of the generated captions.
- **Dataset Scarcity:** The biggest Bengali dataset, BNATURE, only has 8,000 images. This is tiny compared to English datasets, making it hard to train deep neural networks or use modern architectures like Transformers and vision-language pre-training.
- **Cultural Contextualization:** Most models are trained on general images and do not understand specific Bengali cultural elements like festivals, traditional clothing, or local landmarks. This makes the captions feel generic and less useful for native speakers.

2.4 Explainable AI in Image Captioning

Explainability is now a must-have for vision-language models, especially when they are used in high-stakes fields like medical imaging, self-driving cars, and assistive tech for the visually impaired. In these areas, accuracy is not enough on its own; we need to know that a model's logic is sound. Explainable AI (XAI) bridges this gap by making model predictions easier to interpret and trust, ensuring they align with how a human would actually think.

2.4.1 Attention Mechanisms as Interpretability Tools

Attention mechanisms were first created to help models focus on specific parts of an image, which improved performance. Over time, they have also become a go-to tool for interpretability. By looking at attention weights, researchers and users can see which parts of an image the model "looked at" to generate certain words. However, using attention for explanations has several serious flaws:

- **Attention Is Not the Same as an Explanation:** Just because a model assigns high weight to a specific area does not mean that area actually caused the model's decision. Attention shows correlation (what the model was looking at), not causation. For example, a model might focus on a comma or a background pixel while still predicting the correct word.
- **Confusion in Multi-Head Attention:** Modern Transformer models use "multi-head" attention, where different "heads" look at different things simultaneously. Trying to combine or pick between these different views to create a single visualization often leads to confusing or misleading results.

- **No Way to Test "What If" Scenarios:** Current attention maps are descriptive, but they are not contrastive. They can show what the model looked at, but they can not tell you how a caption would change if you altered a specific part of the image. This missing "counterfactual" reasoning is a major hurdle for truly understanding how these models think.

Even with these problems, attention remains a basic building block for explainable systems and serves as a starting point for more advanced research.

2.4.2 Attribution Methods and Saliency Maps

Gradient-based attribution methods, such as Grad-CAM, SmoothGrad, and Integrated Gradients, offer another way to look at explainability by measuring how sensitive model outputs are to specific input features. These techniques create saliency maps that pinpoint the exact parts of an image that had the most influence on a caption.

Compared to relying solely on attention, attribution methods provide a few distinct advantages:

- **Causal Insight:** These techniques measure how changing input pixels actually alters the output, which gives a clearer idea of what is truly driving the model's decision.
- **Flexibility:** You can use attribution methods on any model that allows for gradient calculations, even if it does not use an attention mechanism.
- **Detailed Localization:** Saliency maps can focus on specific pixels or small regions, offering a much more detailed look at how the model processes information.

That said, these methods are not perfect. They can be overly sensitive to small setting changes, vulnerable to being tricked by adversarial inputs, and can still struggle to perfectly separate simple correlation from true causation.

2.4.3 Region-Word Alignment and Cross-Modal Grounding

Region-word alignment techniques create a direct link between specific parts of an image; like detected objects or certain spatial areas as well as the words in a caption. These methods generally follow a three-step process:

- **Object Detection:** First, models like Faster R-CNN or YOLO identify and locate various objects within the image.
- **Cross-Modal Matching:** Next, the system calculates alignment scores to match these detected regions with specific words in the caption, often using contrastive learning or matching mechanisms.
- **Visualization:** Finally, the system shows its work by overlaying bounding boxes or heatmaps on the image, labeled with the specific words they represent.

This approach offers clear, easy-to-understand explanations. It works especially well in fields that need precise detail, such as medical reporting or visual question answering. However, it does come with downsides: it requires more computing power to run the object detection, and if the detector misses or mislabels an object, the final explanation will be wrong.

2.5 Identified Research Gaps

The literature review highlights three major gaps that current research has not addressed:

Gap 1: No Bengali-Optimized Explainable Systems. Existing Bengali captioning models show decent results, but they do not explain how they reach their conclusions. Meanwhile, most explainable systems are built for English. These do not work well for Bengali because they can not handle the unique script, complex word structures, or specific cultural context of the language.

Gap 2: Disconnection Between Explanations and Actual Users. Research into explainable AI usually creates tools for researchers, like heatmaps or complex data maps. There is very little focus on making these explanations useful for everyday people, such as the visually impaired or students. We still have not seen these explanations integrated into helpful tools like voice commands or text-to-speech.

Gap 3: A Lack of Ready-to-Use Frameworks. Most studies focus on academic prototypes tested on standard datasets. They rarely turn into practical, user-friendly software with simple interfaces or real-time processing. This gap between a lab experiment and a real-world application makes it hard for people to actually benefit from the technology, especially in regions where Bengali is the primary language and resources are limited.

2.6 Bibliometric Findings

Bibliometric summaries show rapid growth in explainable image captioning (notably after 2020), concentrated in conference venues and a small set of active research groups. Bengali-focused works are rare, reinforcing the need for targeted dataset and model development.

2.7 Recommendations for Future SLRs and Studies

Future reviews should report full search protocols, apply dual independent screening as well as extracted datasets. For Bengali captioning, researchers should collect multiple captions per image, adopt morphology-aware tokenization, and combine attention with attribution and region-word alignment for richer explanations.

3 Methodology

3.1 Motivation and Purpose

The development of a comprehensive dataset for Bangla image captioning addresses a clear gap in multimodal resources for the Bangla language. Existing large-scale image-captioning corpora are often English-dominated or culturally distant, which limits the performance of models on Bangla-specific visual contexts. While several Bangla datasets exist, many are limited in scope, domain coverage, or vocabulary and therefore do not fully capture the visual diversity present across Bangladesh.

To fix these issues, we put together a varied dataset that includes several different categories, such as people, animals, objects, scenes from nature, and everyday activities. This dataset is designed to help train better models and provide more culturally accurate testing for Bengali image captioning research.

3.2 Dataset Sources

3.2.1 Training Dataset Composition

The training dataset includes 18,539 validated images with Bengali captions, collected from multiple established sources and supplemented with custom data to fill gaps and increase variety. The dataset covers human activities, natural environments, everyday objects, and categories that typically get less attention. Table 3.1 shows the training dataset structure and caption statistics across all sources.

Table 3.1: Training Dataset Composition and Caption Statistics

Dataset Source	**Images**	**Captions/Image**	**Total Captions**
BAN-Cap	8,091	5	40,455
BanglaLekhaImageCaptions	9,154	2	18,308
Bangla Image Captioning	1,146	5.68 (avg)	6,509
Custom Dataset	148	5	740
Total	**18,539**	**3.54 (avg)**	**65,912**

Training Dataset Source Descriptions

- **BAN-Cap (8,091 images, 40,455 captions):** This is the largest part of the training set, focusing on images of people and their activities. Every image has five captions, which helps show different ways to describe the same scene in Bengali. Since it centers on actions and how people interact, it serves as the base for the model's understanding of human activity.

- **BanglaLekhaImageCaptions (9,154 images, 18,308 captions):** This source has the most images, featuring objects, indoor settings, and everyday items. Each image comes with two captions to offer different perspectives. It balances out the human-focused data from BAN-Cap by teaching the model how to describe objects and spatial layouts.
- **Bangla Image Captioning (1,146 images, 6,509 captions):** This dataset covers wildlife, landscapes, and nature. It has the most captions per image; averaging 5.68—because natural scenes often need more descriptive variety. This ensures the model learns the right vocabulary for outdoor environments and different animals or plants.
- **Custom Dataset (148 images, 740 captions):** We built this small, specific collection to fill in gaps found during our initial analysis. Like BAN-Cap, it uses five captions per image. It targets specific cultural details and visual categories that the other datasets missed, making the model more reliable across different situations.

3.2.2 Testing Dataset Composition

To see how well our model works on completely new data, we used a separate testing set from two different sources: **BNATURE** and **BNLIT**. We did not use any of this data during training or validation, so the evaluation is a fair test of how the model handles information it has never seen before. Once training was finished, we took the model checkpoint from epoch 3 and ran it through a specific evaluation process designed for this test set.

Table 3.2 shows the testing dataset structure and validation statistics.

Table 3.2: Testing Dataset Composition and Validation Statistics

Source	Images	Validated	Captions/Image	Total Captions
BNATURE	8,000	1,000	5.00 (avg)	5,000
BNLIT	8,743	500	1.00	500
Total	**16,743**	**1,500**	**3.67 (avg)**	**5,500**

Testing Dataset Source Descriptions

BNATURE Test Subset (1,000 images, 5,000 captions): We pulled this subset from a larger collection of 8,000 images, specifically selecting 1,000 that passed a full check for file integrity and caption availability. These images mostly show nature, wildlife, and outdoor settings. Most images have five captions, though some have as many as seven, giving us a solid range of reference points to evaluate the model's performance. During our check, we confirmed all 1,000 images and their captions were present and accounted for; we left out the other 7,000 images because they lacked proper caption references.

- **Varying Caption Counts:** While five captions is the standard here, some images have up to seven, which helps us see how the model handles more detailed descriptions.
- **Verified Data:** We made sure there were no missing files or broken links, so the test results are reliable.

- **Focus:** The content stays centered on natural scenes, animals, and landscapes.

BNLIT Test Subset (500 images, 500 captions): This subset includes 500 validated images chosen from a broader collection. It's a bit different because it focuses on Bengali culture and literary contexts, with each image having just one caption. While the full BNLIT collection is much larger, many files were missing or lacked captions, so we stuck to these 500 fully verified images for our testing.

- **Single-Caption Testing:** Since there's only one reference caption per image, this tests if the model can hit a specific target description rather than just being "close enough."
- **Cultural Depth:** These images often show traditional scenes and artifacts, which pushes the model to understand specific cultural nuances.
- **Organized Structure:** The files were simple and numbered sequentially which made the evaluation process straightforward.

Testing Dataset Statistics Overview

The combined testing dataset provides a rigorous evaluation framework:

- **Total validated test images:** 1,500 images across two domains (natural scenes and cultural contexts)
- **Total test captions:** 5,500 captions (mean: 3.67 captions/image; median: 1.0 because BNLIT uses single captions)
- **Caption length:** 2–22 tokens (mean: 6.37; median: 6.0; std: 2.90), slightly shorter than training captions (mean: 8.44), showing more concise test references
- **Complete validation:** All 1,500 images have verified file existence and caption availability with zero missing or corrupted entries
- **Distinct vocabulary:** 2,805 unique tokens in test captions, about 12.6% of the training vocabulary (22,276 tokens), so evaluation includes both common and rare words

Evaluation Methodology

We evaluated the model using a specific workflow that was separate from the training process to ensure the results were accurate and unbiased.

1. **Model Selection:** We used the checkpoint from epoch 3, which had the best validation loss (0.9367). This was loaded into a dedicated evaluation setup built specifically to test data the model had not seen yet.
2. **Caption Generation:** For each of the 1,500 test images, the model created one Bengali caption. We used the same method as training; beam search decoding with a width of 3; to keep things consistent.
3. **Comparing Results:** We checked the model's captions against every available reference. For BNATURE images, that meant comparing against five different captions; for BNLIT, it was a one-to-one comparison. We used standard scoring methods like BLEU and ROUGE to measure accuracy.

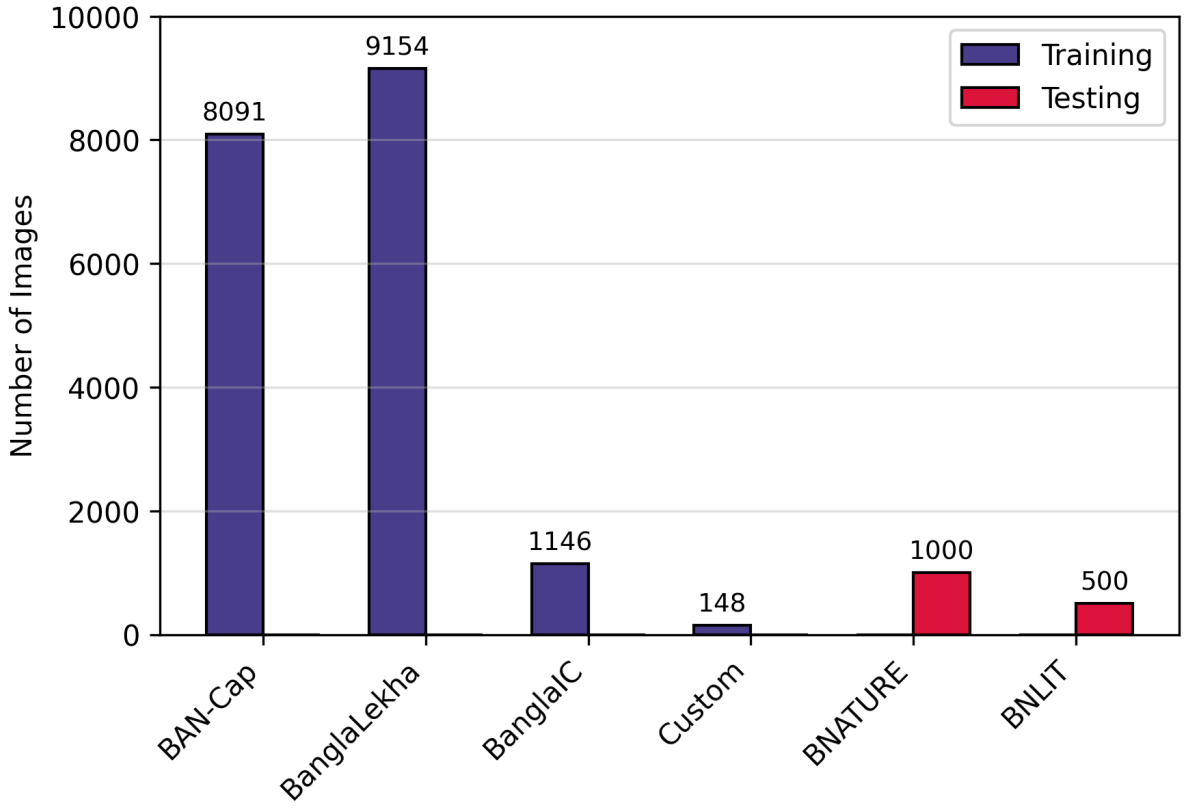


Figure 2: Training and Testing Dataset Composition

4. **Final Analysis:** We gathered the scores for every image to find the average and median performance. We also looked at the BNATURE and BNLIT results separately to see how the model handled multi-reference versus single-reference tasks.

By keeping the training and testing data completely separate and using a dedicated evaluation process, we can be sure these scores show how well the model actually works in the real world, rather than just how well it memorized the training data.

3.2.3 Caption Annotation and Quality Assurance

To ensure the text was both linguistically sound and consistent, every caption in the training and testing sets went through a human review process. We followed standard Bengali grammar and style rules to maintain high quality across the board.

1. **Native Speaker Annotation:** All captions were written by native speakers. This was vital to ensure the language felt natural and remained culturally relevant to the Bengali-speaking community.

2. **Grammatical Validation:** Experts reviewed the captions for proper grammar and correct use of the Bengali script. This included a careful check of complex conjuncts and diacritics; unique parts of the script that are often difficult for automated systems to handle correctly.

3. **Semantic Consistency:** We cross-checked multiple captions for the same image. The goal was to make sure they all accurately described the scene while using a variety of words and sentence structures.

4. **Quality Control:** We cleaned the data by removing any images with missing or corrupted files. For the training set, we excluded 36 images and removed 549 unused ones; for the testing set, we confirmed that all 1,500 images had full, verified caption coverage.

The final collection includes 71,412 captions across 20,039 images. Because of this rigorous process, the dataset offers a high level of linguistic diversity and reliable quality for both training and testing.

3.2.4 Vocabulary and Linguistic Statistics

Training Dataset Vocabulary

The training dataset shows a strong linguistic range with a vocabulary of **22,276 unique tokens**. This variety allows the model to accurately describe a wide array of visual concepts, from specific objects and actions to more subtle environmental details.

Our analysis of the captions reveals a clear focus on concise, direct descriptions:

- **Caption Length:** Most captions fall between 7 and 12 tokens, with an overall average of 8.44 tokens per description.
- **Range and Distribution:** While the shortest captions are just 2 tokens, some reach up to 38, allowing for more detailed context when a scene is particularly complex.
- **Consistency:** With a median length of 8.0 tokens and a standard deviation of 3.61, the data is consistently structured, ensuring the model learns to provide balanced descriptions without being overly wordy.

The combination of a broad vocabulary and focused sentence lengths helps the model generate Bengali captions that are both descriptive and easy to understand. Table 3.3 shows the 20 most frequent tokens in the training dataset.

Table 3.3: Top 20 Most Frequent Tokens in Training Dataset

Token	English	Freq.	Token	English	Freq.
একটি	a/an (inanimate)	32,974	করছে	doing/is doing	4,431
একজন	a/an (person)	18,024	দাড়িয়ে	standing	4,389
আছে।	is/are (sentence-final)	10,541	উপর	on/above	4,242
আছে	is/are	9,722	কালো	black	3,973
লোক	person/people	8,833	নিয়ে	with/having	3,953
কুকুর	dog	7,902			
বসে	sitting	7,188			
এবং	and	6,646			
এক	one	6,447			
পুরুষ	man/male	5,298			

Testing Dataset Vocabulary

The testing dataset vocabulary has **2,805 distinct tokens**, about 12.6% of the training vocabulary. This smaller vocabulary makes sense given fewer captions (5,500 vs. 65,912) and reflects focused coverage of natural and cultural contexts. Caption lengths in the testing dataset are slightly shorter: 2–22 tokens (mean: 6.37; median: 6.0; std: 2.90) compared to training data, showing that test references tend to be more brief. Table 3.4 shows the 20 most frequent tokens in the testing dataset.

Table 3.4: Top 20 Most Frequent Tokens in Testing Dataset

Token	English	Freq.	Token	English	Freq.
আছে	is/are	2,472	লোক	person/people	371
একজন	a/an (person)	2,302	জন	person (classifier)	348
একটি	a/an (inanimate)	1,858	মেয়ে	girl	347
মানুষ	person/human	1,071	শিশু	child	333
দাড়িয়ে	standing	822	যাচ্ছে	going	296
বসে	sitting	795	নৌকা	boat	278
আছে।	is/are (sentence-final)	752	করছে	doing	273
ছেলে	boy	732			
পুরুষ	man/male	597			
কিছু	some/something	539			
দিয়ে	with (instrumental)	492			
ও	and/also	434			
নিয়ে	with/having	426			

Vocabulary Overlap Analysis

A comparison between the training and testing sets reveals a strong overlap in high-frequency words. This confirms that the model is being evaluated on consistent Bengali patterns rather than random or unfamiliar data.

- **Grammatical Foundations:** Both datasets share the same core linguistic markers. This includes copular verbs like আছে (ache, meaning "is" or "has"), indefinite articles like একজন (ekjon for people) and একটি (ekti for objects), and postpositions like দিয়ে (diye) and নিয়ে (niye). These words appear at similar rates in both sets, proving the sentence structures remain consistent.

- **Human-Centric Focus:** There is a clear overlap in words used to describe people and actions. Common tokens in both lists include মানুষ (people), লোক (person), পুরুষ (man), ছেলে (boy), মেয়ে (girl), and শিশু (child). Actions like দাঁড়িয়ে (standing), বসে (sitting), করছে (doing), and যাচ্ছে (going) are also prominent across both datasets.

- **Domain-Specific Differences:** The testing set introduces নৌকা (nouka, or "boat") as a top-20 word, appearing 278 times. This reflects the BNATURE subset's focus on river and water scenes. While this word exists in the training data, it is far more common in the test set.

- **Subject Balance:** The test set is more focused on people than the training set. In the testing data, the ratio of the "person" article (একজন) to the "object" article (একটি) is 1.24:1, whereas in the training data, it is 0.55:1. This shift aligns with the specific content found in the BNATURE and BNLIT subsets.

This overlap means the 2,805 unique words in the test set act as a representative sample of the much larger training vocabulary. It ensures the evaluation measures how well the model actually understands Bengali rather than penalizing it for missing rare or unrelated words.

3.3 Preparing the Dataset for Model Training

We performed a number of preparation steps prior to training. Key actions include:

- Fixing misaligned captions in the BanglaLekhaImageCaptions subset so that each caption correctly matches its image.
- Structuring the custom dataset into category-based subfolders to simplify sampling and analysis.
- Producing explicit search queries for each collection target to enable reproducible image collection for underrepresented subcategories.

The search queries used for the custom collection emphasize realistic, region-specific terms relevant to Bangladesh. Representative queries by category are listed below in Table 3.5.

Table 3.5: Image Query Prompts by Category for Bengali Cultural Context in Custom Dataset

Category & Label	**Generation Prompt**
Vehicle	
Rickshaw	Rickshaw on street in Dhaka Bangladesh
CNG	CNG auto rickshaw Bangladesh
Bus	Public bus Bangladesh crowded road
Car	Bangladeshi private car in urban area
Bicycle	Bangladeshi boy riding bicycle village
Bike	Motorbike on rural road Bangladesh
Human	
Boy	Young Bangladeshi boy playing outside
Girl	Bangladeshi girl wearing sari or borka portrait
Man	Bangladeshi man wearing Punjabi or lungi street photo
Woman	Bangladeshi woman in sari or borka walking
People	Group of Bangladeshi people crowd urban village
Group	Friends group photo in Bangladesh rural setting
Child	Bangladeshi children group playing in village area
House	
Broken_House	Broken old house Bangladesh flood affected
Good_House	Modern house in rural Bangladesh village

Continued on next page

Table 3.5 – *Continued from previous page*

Category & Label	Generation Prompt
Object and Place	
Road	Village road with people and vehicles Bangladesh
Market	Busy rural market in Bangladesh
School	Government school building and students Bangladesh
Animal	
Cow	Cow grazing in Bangladeshi field
Goat	Goat near house in Bangladesh
Dog	Stray dog resting on street Bangladesh
Cat	Domestic cat sitting in Bangladesh home
Horse	Horse pulling cart in Bangladesh
Bird	Common birds flying in Bangladesh
Insect	Close-up insect Bangladesh nature
Flower	
Flower	Various flowers blooming Bangladesh
Big_Flower	Close-up big flower Bangladesh
Small_Flower	Tiny garden flowers Bangladesh
Red_Flower	Red hibiscus rose flower Bangladesh
Unknown_Flower	Uncommon exotic flowers Bangladesh
Fruit	
Fruit	Fruit shop or basket in Bangladesh
Guava	Fresh guava tree Bangladesh
Banana	Banana bunch hanging tree Bangladesh
Jackfruit	Jackfruit on tree Bangladesh rural area
Mango	Mango harvesting season Bangladesh
Orange	Orange fruit Bangladesh tree
Nature	
Tree	Single tree in green paddy field Bangladesh
Environment	Clean village environment in Bangladesh
River	Calm river flowing through rural Bangladesh
Blue	Blue sky over rice field Bangladesh
Sky	Sky in countryside Bangladesh
Water	Clean pond or river water Bangladesh
Gestures	
Lying	Person lying on grass Bangladesh village
Sitting	Bangladeshi man or child sitting outdoor
Standing	Bangladeshi woman standing wearing sari

3.3.1 Category Rationales and Query Choices

We carefully selected these categories to ensure the model understands the specific visual and cultural landscape of Bangladesh. Each group of images addresses a different part of daily life, from urban transport to rural traditions. (see Table 3.5 for the full list of prompts used to gather these images).

Vehicle: Since vehicles like rickshaws and CNGs are everywhere in Bangladesh, they are vital for teaching the model about motion and urban context. We used queries like "Rickshaw on street in Dhaka" to capture these unique transport types.

Human: Capturing local clothing (like saris or lungis) and social activities is essential for culturally accurate captions. Search queries focused on diverse portraits and street photos of people across different ages.

House: We included built environments to help the model recognize socioeconomic conditions, such as the difference between modern rural homes and houses damaged by floods.

Object and Place: This category provides the "anchors" for a scene; busy markets, village roads, or government schools which help the model describe complex public spaces.

Animal: Domestic and wild animals are common in both rural and peri-urban areas. We chose queries like "Cow grazing in field" or "Stray dog on street" to reflect these everyday sightings.

Flower & Fruit: These add visual variety and seasonal context. Queries for jackfruit or mango harvesting help the model learn domain-specific vocabulary related to Bangladesh's agriculture and nature.

Nature: Landscape images of paddy fields and rivers help the model master spatial relationships and general outdoor descriptions.

Gestures: By focusing on specific poses like someone sitting outdoors or standing in a sari; we help the model distinguish between different actions and interactions, leading to more detailed descriptions.

3.4 Dataset Structure and Core Features

This dataset is built to help models learn how to connect visual information with the Bengali language. It covers a wide range of images and human-written descriptions, focusing on clear details and cultural accuracy.

3.4.1 Diversity of Visual Categories

To help the model work well in different situations, the dataset includes a variety of categories:

- **Humans and Group Interactions:** Capturing daily activities and social scenes.
- **Animals:** Featuring both domestic and wild animals in their natural settings.
- **Everyday Objects:** Helping the model identify common items in urban and rural environments.
- **Scenes and Landscapes:** Including busy streets, village markets, and various natural outdoor settings.

3.4.2 Caption Characteristics

The captions are designed to be concise and accurate rather than overly flowery.

- **Focus:** Descriptions prioritize identifying objects, explaining what is happening, and noting where things are in the scene.
- **Vocabulary:** With a unique word count of 22,276, the dataset offers a rich linguistic base for the model to learn from.
- **Length:** Captions are brief, averaging about 8 to 9 words, which helps the model stay focused on the main subjects.

3.5 Pipeline and Operational Details

Our project follows a structured workflow that handles everything from raw data to a working application. Here's a breakdown of how the system is built and maintained:

3.5.1 Data and Model Training

- **Data Preparation:** We normalize, resize, and tokenize all images and text to keep the data consistent. Every step of this preprocessing is recorded so we can replicate the results exactly if needed.
- **Model Architecture:** The system uses an encoder-decoder with attention. Essentially, the encoder looks at the image to identify key features, while the decoder "translates" those features into words. The attention mechanism acts as the MVP here; it allows the model to focus on specific parts of an image as it generates each word, much like how a human looks at a person before describing their clothes.
- **Training Progress:**We use learning rate (LR) schedules to adjust how quickly the model learns over time. We also save regular "checkpoints" to log our progress (using metrics like BLEU) and ensure we do not lose work if training is interrupted.

3.5.2 Pipeline Deployment

- **The Service:** The final model is hosted as an API service. Users can upload an image, and the system returns a Bengali caption. The system can even provide attention visualizations; maps that show exactly which parts of the image the model was looking at when it chose certain words.
- **Infrastructure:** To keep the setup portable and easy to run, we use containers. These act as isolated environments that hold all the necessary code and libraries. For faster performance, we use GPU-enabled versions of these containers to speed up the heavy processing required for image analysis.

3.5.3 Quality Check

- **Evaluation:** We run the model against a "held-out" test set (data it has never seen before) to see how well it actually performs. This batch inference process gives us a realistic look at the model's accuracy through detailed per-example reports.



Figure 3: Model Training, Evaluation, and Deployment Workflow Pipeline

3.5.4 Experimental Setup

Training for this project was carried out on a single workstation configured for GPU-accelerated deep learning. The workstation runs Windows 11 Pro on an ASUS PRIME H770-PLUS motherboard equipped with a 13th-generation Intel Core i7-13700K (16 cores, 24 logical processors), 64 GB of system RAM, high-performance NVMe storage, and an NVIDIA GeForce RTX 4060 Ti GPU used for CUDA-accelerated training. Experiments were executed with GPU acceleration to improve throughput; training jobs were monitored with structured logging and automatic checkpointing, and early stopping based on validation loss was employed to avoid overfitting. All runtime environments, dependency lists, hyperparameters, and random seeds are recorded in the project's environment specifications and the checkpoint manifests; artifacts (model weights, tokenizer, preprocessing config) are versioned and checksumed to ensure exact reproducibility of reported results.

Table 3.6: Training workstation hardware summary

Component	Specification
CPU	Intel Core i7-13700K (13th Gen; 16 cores, 24 threads)
GPU	NVIDIA GeForce RTX 4060 Ti (CUDA-enabled)
Memory	64 GB RAM
Storage	High-performance NVMe SSD
OS	Windows 11 Pro
Network	Gigabit Ethernet
Notes	CUDA-accelerated training

3.5.5 Real World Deployment

We moved the model into a practical, live environment as a lightweight service. It works through a simple API where users can upload an image and receive a Bengali caption, sometimes paired with "attention visualizations" that show exactly what part of the image the model focused on.

To test this in the field, we used the Orange Pi Zero3 [22], a small but capable single-board computer (SBC), as seen in Figure 4. Our specific setup uses a version with 1.5 GB of RAM and an Allwinner H618 quad-core CPU.

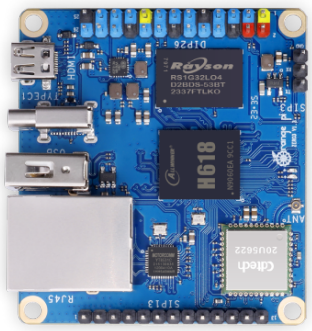


Figure 4: Orange Pi Zero3 (Image Source: [1])

Deployment Strategy for the Orange Pi Zero3 [22]

Running a vision model on a small device requires a clean, organized setup:

- **Containerized Service:** The model runs in its own "container," a self-contained environment that keeps the software stable and isolated. This makes it easy to update the system or roll back to a previous version if something goes wrong.
- **External Storage:** Because the Orange Pi has limited space, we do not pack the heavy model files directly into the software. Instead, we keep the model weights and settings on external storage and "mount" them so the system can access them whenever it needs to.
- **Security & Monitoring:** We check every uploaded file to make sure it is valid and keep the device on a local network to reduce security risks. The service also keeps logs to ensure everything is running smoothly.
- **Smooth Updates:** When we have a new version of the model, we test it on a different machine first. Once we know it works, we push it to the device and restart the service.

Why Use a Small Single-Board Computer? While a full desktop is faster, we chose the Orange Pi Zero3 [22] for several practical reasons:

- **Portability & Power:** These devices use very little energy; sometimes as low as 0.8W to 2.2W. This means they can run for a long time on a standard power bank, making them perfect for areas where electricity is unreliable.
- **Cost-Effective:** Using low-cost hardware like the Orange Pi helps us prove that high-tech tools like DrishtiXAI do not need expensive equipment to work.
- **Efficiency:** Working within the limits of a small device forced us to optimize our code. By focusing on efficiency, we ensured the model is fast and reliable even when resources are tight.

3.5.6 Operational Considerations

We followed a reproducible, production-oriented workflow when running and maintaining the pipeline. In particular, we:

- Ensured strict compatibility between artifacts and preprocessing by versioning and pairing each checkpoint with its tokenizer and preprocessing manifest.
- Managed large artifacts by mounting checkpoints or using external artifact storage rather than embedding heavy files in container images.
- Maintained operational visibility via structured logging and monitoring to support auditing and debugging.
- Enforced input validation and basic security controls at the inference boundary and archive environment manifests (library versions, config) to guarantee reproducibility of results.
- Reduced deployment latency and failure risk with automated artifact verification (checksums and shape checks), and clear rollout/rollback procedures.

3.6 Model

3.6.1 Model Description

Backbone Selection. **InceptionV3** is used as the primary convolutional backbone in our experiments. We also evaluated **EfficientNetB4** as an alternative backbone; however, in our trials it produced captions that were often irrelevant to the image content and therefore was not adopted for the main experiments.

Implementation Details of the InceptionV3 Experiment

The InceptionV3 experiment is implemented in TensorFlow (`tf.keras`) using a set of modular, subclassed models: `CNN_Encoder`, `BahdanauAttention`, and `RNN_Decoder`. The overall design closely follows the *Show–Attend–Tell* framework, with the following concrete architectural and engineering choices (all parameter values correspond to those used in the provided notebook).

- **Vision backbone:** A pretrained **InceptionV3** network is used with `include_top = false` and `weights = 'imagenet'`. The output of the final convolutional layer has spatial dimensions $8 \times 8 \times 2048$, which is reshaped into a sequence of $K = 64$ spatial feature vectors of dimension 2048, yielding a tensor of shape $(B, 64, 2048)$.
- **CNN encoder (projection head):** A lightweight projection module applies a fully connected layer with output dimension **EMBED_DIM** $= 256$ and ReLU activation independently to each spatial feature vector. This produces encoder features
$$\mathbf{F} \in \mathbb{R}^{B \times K \times d'}, \quad d' = 256,$$
which serve as the input to the attention mechanism.
- **Attention mechanism:** Additive (Bahdanau) attention is implemented, with the attention unit size in `BahdanauAttention` equal to the decoder GRU hidden size (**units** $= 512$). At each decoding timestep t, the attention scores, weights, and context vector are computed as:
$$\begin{aligned} e_{t,i} &= \mathbf{v}^\top \tanh\left(W_1 \mathbf{f}_i + W_2 \mathbf{h}_{t-1}\right), \\ \alpha_{t,i} &= \mathrm{softmax}_i(e_{t,i}), \\ \mathbf{c}_t &= \sum_i \alpha_{t,i} \mathbf{f}_i, \end{aligned}$$
where $\mathbf{f}_i$ denotes the i-th spatial feature and $\mathbf{h}_{t-1}$ is the previous decoder hidden state. The attention layer returns both the context vector $\mathbf{c}_t$ and the attention weights $\alpha_{t,\cdot}$ for visualization.
- **Decoder (`RNN_Decoder`):** The decoder consists of the following components:
  - An `Embedding` layer with vocabulary size **vocab_size** = `top_k` + 1 (with `top_k` = 10000) and embedding dimension **embedding_dim** = 256.
  - A `GRU` layer with **units** = 512, configured with `return_sequences = True` and `return_state = True`, and using `glorot_uniform` initialization.
  - A projection head composed of two Dense layers that map GRU outputs to vocabulary logits (the final Dense layer produces unnormalized logits).

At each timestep, the decoder concatenates the context vector $\mathbf{c}_t$ with the embedded input token $\mathbf{x}_t$ and feeds the result into the GRU. During training, teacher forcing is applied. During inference, captions are generated using stochastic sampling via `tf.random.categorical` (instead of greedy argmax), and decoding terminates when the special `<end>` token is produced.

- **Tokenizer and vocabulary:** Text tokenization is performed with `top_k` = 10000 and `oov_token = '<unk>'` using `tf.keras.preprocessing.text.Tokenizer` . The padding token `'<pad>'` is explicitly assigned index 0. The tokenizer is serialized and stored as `tokenizer.pkl` to ensure reproducible decoding and evaluation.

- **Loss, optimizer, and training loop:** The training objective with `from_logits = True` and `reduction = 'none'` uses `SparseCategoricalCrossentropy`. Per-timestep losses are masked to ignore padding tokens (pad index = 0). Optimization is performed using the **Adam** optimizer. The training loop precomputes InceptionV3 features and stores them as `.npy` files, constructs `tf.data` pipelines to load image features and caption sequences, and performs explicit training steps using `tf.GradientTape`. Model checkpointing is handled via `tf.train.CheckpointManager`, maintaining both regular and best-performing checkpoints along with a small JSON metadata file describing the best model.

- **Hyperparameters and practical settings:**
  - Image resolution: $(299, 299)$ (InceptionV3 input size)
  - Vocabulary size parameter: **top_k** = 10000
  - Embedding dimension: **embedding_dim** = 256
  - GRU and attention units: **units** = 512
  - Batch size: **BATCH_SIZE** = 64
  - Feature dimension: 2048; spatial locations $K = 64$
  - Number of epochs: **EPOCHS** = 100 with early stopping patience of 5

- **Inference and outputs:** The inference procedure produces both the predicted token sequence and an attention weight matrix of shape (`max_length`, 64). Attention visualizations are generated using Matplotlib and Pillow, with Bengali font overlays to enable qualitative inspection of word-level attention alignment.

- **Reproducibility considerations:** The tokenizer, model checkpoints, per-epoch CSV logs, and best-model metadata are all persisted to disk, enabling exact reloading and reproduction of the trained experiment.

3.6.2 Model Architecture of InceptionV3

We use InceptionV3 as the core of our system to turn images into a grid of features. This setup follows an encoder-decoder path designed to translate visual data into clear Bengali sentences. The architecture has been summarized in Figure 5.

How the System Works:

- **The Encoder:** First, the system takes the input image and extracts specific Region of Interest (ROI) features. These are passed through an ROI encoder that uses standard layers (Dense and ReLU) to prepare the visual data for the next step.

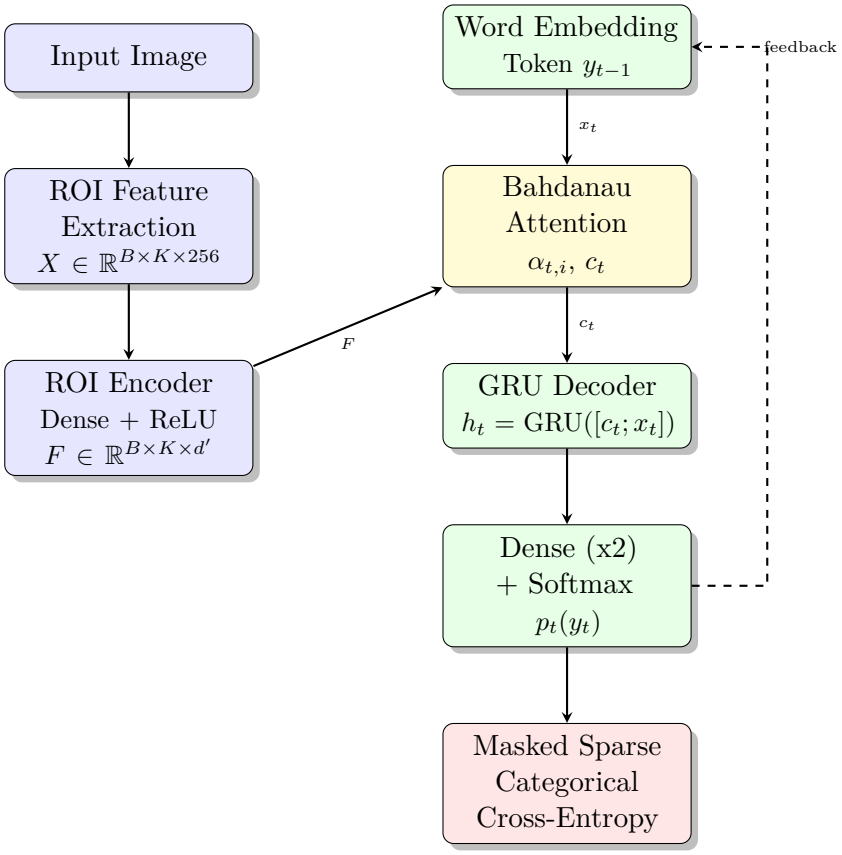


Figure 5: Encoder–Decoder Architecture with Bahdanau Attention for Bengali Image Captioning

- **The Attention Mechanism:** We use Bahdanau Attention to act as a bridge between the image and the text. This allows the model to focus on specific parts of the image while it decides which word to generate next.
- **The Decoder:** The decoder side starts with word embeddings. A GRU decoder takes the context from the attention layer and the previous word to predict the next one. This leads to a final output layer (Dense + Softmax) that chooses the most likely Bengali word.
- **Learning:** The model improves by measuring its performance against the actual captions using a specific loss function (Masked Sparse Categorical Cross-Entropy (SCCE)).

Dataset and Language Details:

- **Training Size:** Our model learned from a set of **18,539 training samples**, with each image paired with several Bengali descriptions.
- **Vocabulary:** The system can choose from **22,276 unique words**, giving it a wide range of language to describe different scenes.
- **Caption Length:** Captions reach up to **38 tokens**, which is long enough to provide detailed context about what's happening in the image.

Notation and Symbols

To make the technical details easier to follow, here is a summary of the mathematical symbols used in our model's architecture:

- **B**: The batch size, or how many images the model looks at during a single training step.
- **K**: The number of image regions (ROIs) per image. For example, an 8×8 grid results in $K = 64$.
- **d**, **d'**: input ROI feature dimensionality and encoder output dimensionality respectively.
- $\mathbf{X} \in \mathbb{R}^{B \times K \times d}$: input ROI feature tensor (per-batch).
- $\mathbf{F} \in \mathbb{R}^{B \times K \times d'}$: encoder-transformed features.
- f_i: the i-th region feature vector (a length-d' vector).
- $\alpha_{t,i}$: attention weight on region i at decoding timestep t (scalar, non-negative, $\sum_i \alpha_{t,i} = 1$).
- c_t: context vector at timestep t, computed as the weighted sum $c_t = \sum_i \alpha_{t,i} f_i$.
- x_t: embedding of the previous token (input to the decoder at time t); $z_t = [c_t; x_t]$ denotes concatenation.
- h_t: decoder hidden state at timestep t; o_t denotes the GRU output used before the final linear layer.
- p_t: probability distribution over the vocabulary at timestep t (output of softmax on logits).
- T: maximum caption length (number of timesteps considered during training/evaluation).
- m_t: binary mask (1 for real tokens, 0 for padding) used to exclude padded positions from loss/metrics.

Encoder: ROI Feature Encoder

The encoder maps Region-of-Interest (ROI) feature tensors into a normalized embedding space. Formally:

$$\mathbf{F} = \mathrm{ReLU}(W_{\mathrm{enc}} \cdot \mathbf{X} + b_{\mathrm{enc}}) \tag{3.1}$$

where $\mathbf{X} \in \mathbb{R}^{B \times K \times d}$ is the ROI feature tensor (with $d = 256$), and $\mathbf{F} \in \mathbb{R}^{B \times K \times d'}$ represents the transformed embeddings.

Attention Mechanism (Bahdanau Attention)

The decoder integrates Bahdanau attention to dynamically weight image features:

$$e_{t,i} = v^\top \tanh(W_1 f_i + W_2 h_{t-1}) \tag{3.2}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_j \exp(e_{t,j})} \tag{3.3}$$

$$c_t = \sum_i \alpha_{t,i} f_i \tag{3.4}$$

Here, c_t denotes the context vector created by attending to salient visual regions.

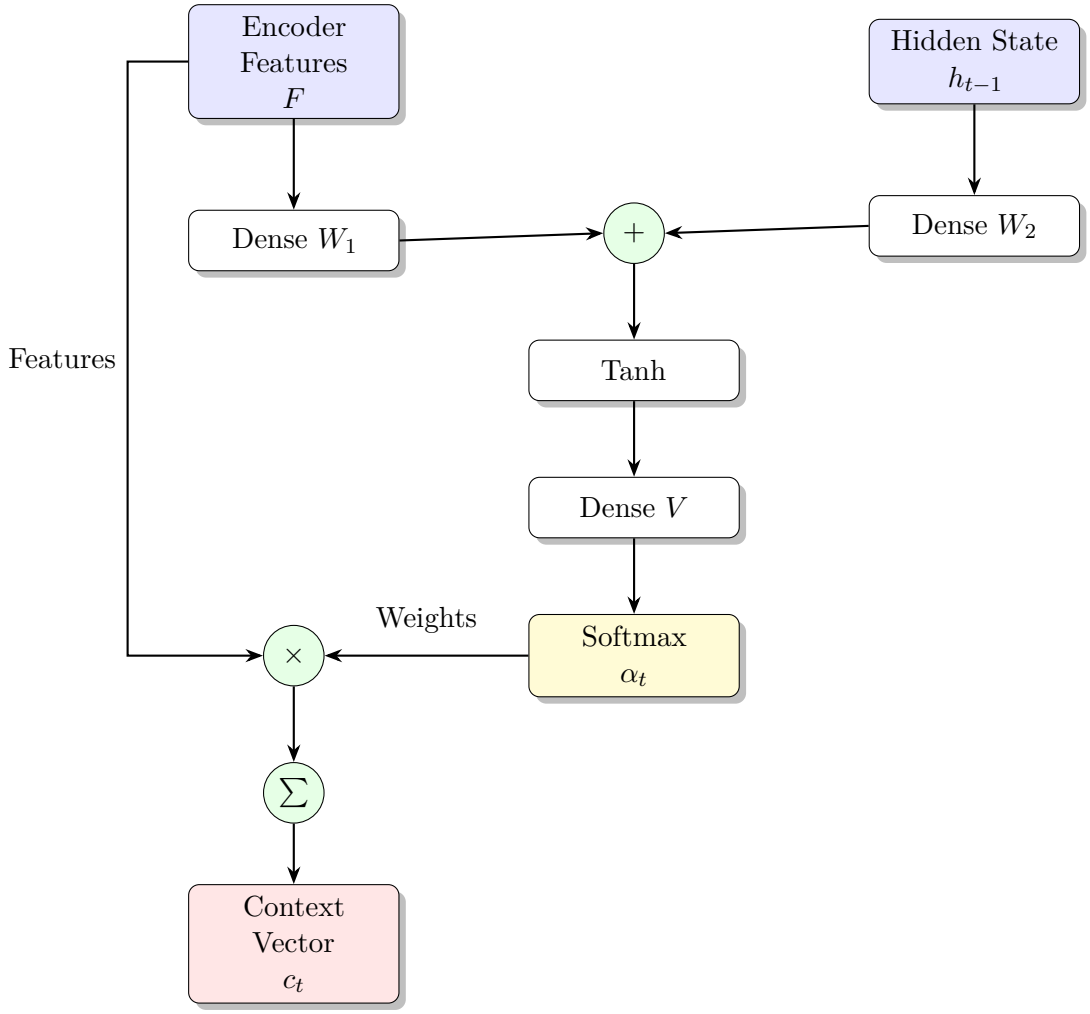


Figure 6: Flowchart of the Bahdanau Attention Mechanism.

Mechanism Description: The Bahdanau attention mechanism computes a context vector c_t that focuses on specific parts of the image relevant to the current word generation. The process is visualized in Figure 6.

1. **Score Computation:** The encoder features F and the previous decoder hidden state h_{t-1} are passed through dense layers W_1 and W_2, respectively. Their outputs are added and passed through a Tanh activation, followed by a final dense layer V to produce unnormalized scores.

2. **Attention Weights:** These scores are normalized using a Softmax function to generate attention weights α_t, which represent the importance of each image region.

3. **Context Vector:** The original encoder features are weighted by α_t and summed together to form the context vector c_t, which is then used by the decoder.

Decoder (GRU-Based Sequence Generator)

Caption generation proceeds via a GRU:

$$x_t = \text{Embed}(y_{t-1}), \quad z_t = [c_t; x_t] \tag{3.5}$$
$$h_t, o_t = \text{GRU}(z_t, h_{t-1}) \tag{3.6}$$
$$a_t = \text{ReLU}(W_a o_t + b_a) \tag{3.7}$$
$$p_t = \text{softmax}(W_o a_t + b_o) \tag{3.8}$$

where p_t is the probability distribution over the next word.

SCCE with Masking

To train the model, we use a specific loss function that measures how far the model's guesses are from the actual captions.

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T} m_t \cdot \log p_t[y_t] \tag{3.9}$$

In simple terms, we calculate the error for every word the model predicts. However, since all captions in a batch need to be the same length, we often have to add "padding" to shorter sentences. We do not want the model to learn from these empty spaces, so we use a mask (m_t). If a token is a real word, the mask is 1; if it is just padding, the mask is 0. This effectively cancels out the padding so it does not affect the final score or the model's training progress.

3.6.3 Model Training

The training process focuses on optimizing the encoder-decoder setup through multiple passes over the data, known as epochs. During each pass, the model updates its internal parameters to close the gap between its predicted captions and the actual Bengali descriptions.

- **Dataset Size:** We used a total of **18,539 image-caption pairs**. Since each image is paired with several descriptions, the model has plenty of examples to help it connect visual scenes with natural Bengali speech.
- **Training Configuration:** We set the training to run for up to **100 epochs**, using a batch size of 64. To keep the process efficient, we used **early stopping**; if the model did not improve for 20 straight epochs, training would end. The training loop also applies a simple learning-rate reduction on plateau (factor = 0.5 after 3 epochs without improvement, with a floor MIN_LR = 10^{-6}). A batch size of $B = 64$ was used while training.
- **Steps per Epoch and Total Iterations:** The number of parameter-update steps per epoch is computed in the training script using integer division: StepsPerEpoch = $\lfloor N_{\text{train}}/B \rfloor$ where N_{train} is the number of training caption samples. For $N_{\text{train}} = 18,539$ and $B = 64$ this yields

$$\text{StepsPerEpoch} = \left\lfloor \frac{18,539}{64} \right\rfloor = 289 \tag{3.10}$$

Hence the nominal total number of parameter-update operations for the configured 100 epochs is approximately:

$$\mathcal{I}_{\text{total}} = \text{StepsPerEpoch} \times \text{EPOCHS} = 289 \times 100 = 289{,}000 \tag{3.11}$$

- **Loss Function:** The learning objective minimizes the *SCCE* loss over all time steps in each generated caption. Given a target sequence of length T, the loss is defined as:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T} \log P(y_t \mid y_{1:t-1}, I) \tag{3.12}$$

  Here, y_t denotes the ground-truth token at step t, and $P(y_t \mid y_{1:t-1}, I)$ represents the probability assigned by the decoder given previous tokens and the encoded image I. We use a mask to ignore any empty padding, ensuring the final score only reflects real words.

- **Optimization Strategy:** We used the **Adam optimizer** to handle the updates. During training, we also used a method called **teacher forcing**, where the model is given the correct word at each step. This helps the system learn the structure of the language much faster in the early stages.

- **Saving Progress:** We checked the validation loss after every epoch and automatically saved the version of the model that performed best. This ensures we do not end up with a model that has simply memorized the training data rather than learning how to describe new images.

- **Time and Environment:** All training was executed within the **CodeOcean environment** running on a workstation, with the full experiment completing in approximately **1 hour, 5 minutes and 6 seconds**. Model weights, tokenizer files, and experiment metadata were archived to support reproducibility and efficient re-initialization.

3.6.4 Training Algorithm

The complete training algorithm is summarized in Algorithm 1.

3.6.5 Model Architecture of EfficientNetB4

EfficientNetB4 is a mobile-friendly yet high-accuracy architecture pre-trained on ImageNet. The model architecture of EfficientNetB4 has been shown in Figure 7. In our experiments the network is loaded with weights from ImageNet and used as a fixed feature extractor (`include_top=false`, frozen). Spatial feature maps are pooled and projected into the common embedding space used by the decoder.

Notes

- **Backbone:** EfficientNetB4 pretrained on ImageNet (chosen for a balance between representational power and compute cost).

- **Pooling Projection:** Spatial features are reduced using GlobalAveragePooling2D and then projected via a small MLP (Dense+ReLU then Dense) into the embedding space of dimension d (EMBED_DIM=768 in our runs).

Algorithm 1 Image Captioning Model Training

Initialize: encoder, decoder, optimizer (Adam), tokenizer, checkpoint managers
Set: best_val_loss = ∞, wait = 0, wait_lr = 0
Hyperparameters: EPOCHS, BATCH_SIZE, patience, LR_REDUCE_FACTOR, LR_REDUCE_PATIENCE, MIN_LR
for epoch = 1 to EPOCHS **do**
    Reset: total_train_loss = 0, total_val_loss = 0
    // Training Phase
    for each batch (img_tensor, target) in training set **do**
        hidden $\leftarrow$ decoder.reset_state(batch_size)
        dec_input $\leftarrow \langle \text{start} \rangle$; features $\leftarrow$ Encoder(img_tensor); loss = 0
        for $t = 1$ to T **do**
            predictions, hidden, _ $\leftarrow$ Decoder(dec_input, features, hidden)
            loss += $\mathcal{L}_{\text{timestep}}(\text{target}_{:,t}, \text{predictions})$ (masked)
            dec_input $\leftarrow \text{target}_{:,t}$ ▷ teacher forcing
        end for
        Compute batch_total = loss/T; apply gradients; total_train_loss += batch_total
    end for
    epoch_train_loss $\leftarrow$ total_train_loss/num_steps
    // Validation Phase
    for each batch in validation set **do**
        Compute validation loss (no gradients); total_val_loss += batch_val_loss
    end for
    epoch_val_loss $\leftarrow$ total_val_loss/val_steps
    // Metrics Computation
    Perform batched greedy decoding on validation set
    Compute metrics: accuracy, precision, recall, F1, ROUGE, token Jaccard
    Save metrics (CSV, plots) and generate report snippets
    // Checkpoint & Learning Rate Management
    if epoch_val_loss < best_val_loss − MIN_DELTA **then**
        best_val_loss $\leftarrow$ epoch_val_loss; wait $\leftarrow$ 0; wait_lr $\leftarrow$ 0
        Save best checkpoint and model summaries
    else
        wait += 1; wait_lr += 1
        if wait_lr $\geq$ LR_REDUCE_PATIENCE **then**
            lr $\leftarrow$ max(lr $\times$ LR_REDUCE_FACTOR, MIN_LR); wait_lr $\leftarrow$ 0
        end if
    end if
    Save regular checkpoint (every 5 epochs)
    if wait $\geq$ patience **then**
        break ▷ early stopping triggered
    end if
end for
Restore best checkpoint; save best-model metadata

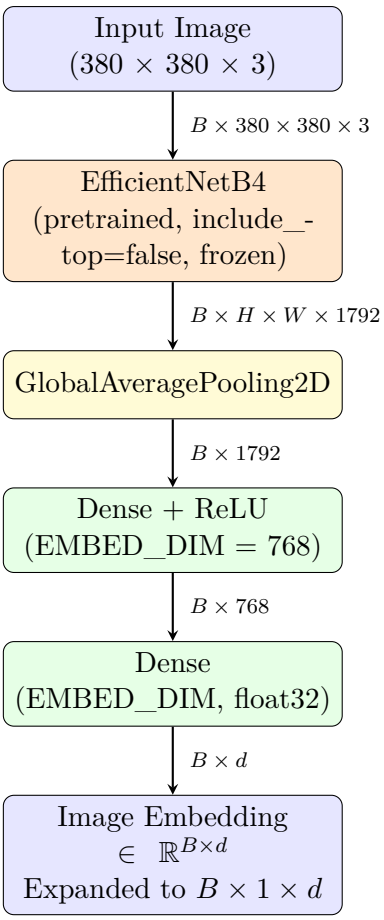


Figure 7: EfficientNetB4 backbone and projection head for computing image embeddings.

- **Frozen Weights:** The backbone weights are kept frozen during captioner training to reduce overfitting and training time; only the projection head and decoder are trained.
- **Integration:** The resulting image embedding is expanded to shape $(B, 1, d)$ and passed to the encoder/decoder (cross-attention) as shown in Figure 5.

3.6.6 Evaluation Metrics

We evaluate generated captions using corpus- and image-level metrics that measure exact matches, token overlap, and n-gram/LCS-based similarity. Let the validation set contain N images. For image index i let $\mathcal{R}_i = \{r_{i,1}, \ldots, r_{i,M_i}\}$ be the set of reference captions and let $\hat{y}_i$ be the single predicted caption produced by the model. Write $\text{tok}(s)$ for the token sequence of a string s, and $|\cdot|$ for the number of tokens.

Notation used in metrics

- N: number of images in the validation set.
- $\mathcal{R}_i = \{r_{i,1}, \ldots, r_{i,M_i}\}$: the set of reference captions for image i.
- $\hat{y}_i$: the model's predicted caption for image i (usually the result of greedy decoding); $\text{tok}(\cdot)$ maps a string to its token sequence and $|\cdot|$ denotes token count.
- ϵ: a small positive constant added to denominators to avoid division-by-zero in metric definitions.
- C_p, C_r: multisets (counters) of tokens in the predicted caption and the chosen reference respectively; $\langle C_p \wedge C_r \rangle$ denotes the sum of per-token minima (i.e., token-wise intersection count).
- $G_n(s)$: multiset of n-grams extracted from string s; $\text{match}_n(\hat{y}_i, r)$ counts clipped matching n-grams between prediction and reference.
- $p_n(\hat{y}_i)$: clipped (modified) n-gram precision used in BLEU (counts in the candidate clipped by the maximum reference counts across $\mathcal{R}_i$).
- c: candidate length $c = |\text{tok}(\hat{y}_i)|$ and r^*: reference length closest to c (used in BLEU's Brevity Penalty (BP) BP).
- BP: brevity penalty in BLEU that penalizes short candidates (see BLEU definition in the following subsection).
- $\text{LCS}(\hat{y}_i, r)$: the length of the LCS between token sequences of the prediction and reference (used in ROUGE-L computations).
- $\text{dist}(a, b)$: Levenshtein (edit) distance between strings a and b (character-level); the normalized Levenshtein ratio is $1 - \text{dist}(a, b)/\max\{|a|, |b|\}$.
- $J(\hat{y}_i, r)$: token-level Jaccard similarity between sets of tokens of the candidate and a reference; when multiple references exist the per-image Jaccard takes the maximum over $r \in \mathcal{R}_i$.

Exact-match Accuracy

The exact-match accuracy counts predictions that exactly match any reference (string equality after trivial normalization):

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\big(\hat{y}_i \in \mathcal{R}_i\big) \tag{3.13}$$

Per-image Token Precision and Recall and Macro Averages

For a given image i we define a per-image token precision P_i and recall R_i by comparing token counts between the predicted caption and a reference. In practice we choose the reference $r_i^* \in \mathcal{R}_i$ that maximizes an F_1 variant (best-match heuristic). Let C_p and C_r be the multisets (counters) of tokens in $\mathrm{tok}(\hat{y}_i)$ and $\mathrm{tok}(r_i^*)$ respectively, and write $\langle C_p \wedge C_r \rangle$ for the count of token-wise intersections. Then

$$P_i = \frac{\langle C_p \wedge C_r \rangle}{|\mathrm{tok}(\hat{y}_i)| + \epsilon}, \tag{3.14}$$

$$R_i = \frac{\langle C_p \wedge C_r \rangle}{|\mathrm{tok}(r_i^*)| + \epsilon}, \tag{3.15}$$

where $\epsilon > 0$ is a small constant to avoid division-by-zero. The macro-averaged precision and recall reported in the paper are the unweighted means over images:

$$\text{MacroPrecision} = \frac{1}{N}\sum_{i=1}^{N} P_i, \tag{3.16}$$

$$\text{MacroRecall} = \frac{1}{N}\sum_{i=1}^{N} R_i. \tag{3.17}$$

An image-level F_1 score can be formed as usual by $F_i = 2P_iR_i/(P_i + R_i + \epsilon)$ and macro-F_1 is the mean of F_i across images.

ROUGE-N

ROUGE–n compares n-gram overlap between prediction and reference. For a prediction $\hat{y}_i$ and a chosen reference r define the multiset of n-grams $G_n(\hat{y}_i)$ and $G_n(r)$. Let

$$\mathrm{match}_n(\hat{y}_i, r) = \sum_{g \in G_n(\hat{y}_i)} \min\big(\mathrm{count}_{\hat{y}_i}(g),\, \mathrm{count}_r(g)\big), \tag{3.18}$$

$$P_n(\hat{y}_i, r) = \frac{\mathrm{match}_n(\hat{y}_i, r)}{|G_n(\hat{y}_i)| + \epsilon}, \tag{3.19}$$

$$R_n(\hat{y}_i, r) = \frac{\mathrm{match}_n(\hat{y}_i, r)}{|G_n(r)| + \epsilon}, \tag{3.20}$$

$$F_n(\hat{y}_i, r) = \frac{2\,P_n(\hat{y}_i, r)\,R_n(\hat{y}_i, r)}{P_n(\hat{y}_i, r) + R_n(\hat{y}_i, r) + \epsilon}. \tag{3.21}$$

When multiple references exist we adopt the common "best-reference" strategy and take the maximum F_n over $r \in \mathcal{R}_i$. The reported corpus ROUGE–n is the mean of per-image best F_n values across $i = 1 \ldots N$.

ROUGE-L (LCS-based)

ROUGE-L uses the length of the LCS between token sequences. Define

$$\mathrm{LCS}(\hat{y}_i, r) = \max\{\, \ell : \ell \text{ is length of a common subsequence between } \mathrm{tok}(\hat{y}_i) \text{ and } \mathrm{tok}(r) \,\}, \tag{3.22}$$

$$P_L(\hat{y}_i, r) = \frac{\mathrm{LCS}(\hat{y}_i, r)}{|\mathrm{tok}(\hat{y}_i)| + \epsilon}, \tag{3.23}$$

$$R_L(\hat{y}_i, r) = \frac{\mathrm{LCS}(\hat{y}_i, r)}{|\mathrm{tok}(r)| + \epsilon}, \tag{3.24}$$

$$F_L(\hat{y}_i, r) = \frac{2\, P_L(\hat{y}_i, r)\, R_L(\hat{y}_i, r)}{P_L(\hat{y}_i, r) + R_L(\hat{y}_i, r) + \epsilon}. \tag{3.25}$$

As with ROUGE–n, we compute per-image ROUGE-L by selecting the reference $r \in \mathcal{R}_i$ that maximizes $F_L(\hat{y}_i, r)$ and report the corpus mean of these per-image F_L scores.

BLEU, Levenshtein Ratio, and Token-level Jaccard

To complement the metrics above we also report the following string- and token-level similarity measures, which are implemented in the evaluation module.

BLEU BLEU measures geometric mean of modified n-gram precisions with a BP to penalize overly short candidates. For a candidate caption $\hat{y}_i$ and reference set $\mathcal{R}_i$ let

$$p_n(\hat{y}_i) = \frac{\sum_{g \in G_n(\hat{y}_i)} \min\big(\mathrm{count}_{\hat{y}_i}(g),\, \max_{r \in \mathcal{R}_i} \mathrm{count}_r(g)\big)}{|G_n(\hat{y}_i)| + \epsilon} \tag{3.26}$$

be the clipped (modified) n-gram precision. With uniform weights $w_n = 1/N$ the sentence-level BLEU–N score is

$$\mathrm{BLEU}_N(\hat{y}_i) = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n(\hat{y}_i)\right), \tag{3.27}$$

where the BP is

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r^*, \\ \exp\big(1 - r^*/c\big) & \text{otherwise,} \end{cases} \tag{3.28}$$

with $c = |\mathrm{tok}(\hat{y}_i)|$ and r^* the reference length closest to c (ties broken by shortest). We use clipped counts across references as shown above.

Levenshtein ratio The Levenshtein (edit) distance $\mathrm{dist}(a, b)$ counts minimum character-level insertions, deletions and substitutions to transform string a to b (computed with dynamic programming). The implementation reports a normalized similarity

$$\mathrm{LevenshteinRatio}(a, b) = 1 - \frac{\mathrm{dist}(a, b)}{\max\{|a|, |b|\}}, \tag{3.29}$$

which equals 1 for identical strings and approaches 0 for very different strings (the implementation returns 1 when both strings are empty).

Token-level Jaccard similarity For token lists $\text{tok}(\hat{y}_i)$ and $\text{tok}(r)$ the token-level Jaccard is

$$J(\hat{y}_i, r) = \frac{|\text{set}(\text{tok}(\hat{y}_i)) \cap \text{set}(\text{tok}(r))|}{|\text{set}(\text{tok}(\hat{y}_i)) \cup \text{set}(\text{tok}(r))| + \epsilon}. \tag{3.30}$$

When multiple references exist we take the per-image value as the maximum Jaccard over $r \in \mathcal{R}_i$, consistent with the best-reference heuristics used for ROUGE and BLEU.

These measures are implemented in the evaluation module and are used alongside ROUGE and exact-match scores to provide a broader view of lexical and character-level similarity between predictions and references.

Implementation notes

All metrics above are implemented using token-level counters and dynamic-programming (for LCS) in the evaluation module. To produce stable, reproducible metric values for validation we run deterministic decoding (greedy argmax) during metric computation; sampling-based decoding can be enabled separately for diversity analysis. Per-epoch metrics are computed on the validation split and stored alongside loss histories to produce the accuracy plot and other visualization figures included in the results directory.

3.7 Mobile Application

The DrishtiXAI mobile application is a cross-platform client (implemented in Flutter) that provides an accessible front-end for the captioning and explanation pipeline. The app targets mobile, desktop, and web platforms and is designed for low-latency interaction, clear accessibility affordances, and straightforward integration with the remote or on-device inference service described in the Deployment section.

3.7.1 Primary Features

The app implements the following primary user-facing features:

- **Image Captioning:** Capture or pick an image and receive a natural-language Bangla caption produced by the inference service.
- **Attention Maps:** Per-token attention visualizations that overlay heatmaps on the image to show where the model focused for each generated word.
- **Color Palette Extraction:** Extraction of primary/secondary/background colors from an image with user presets for UI customization.
- **TTS**: Speak captions to support visually impaired users.
- **Localization and Accessibility:** UI text localized in Bengali for easy understanding while making it configurable for other languages as well through ARB files, clear feedback through snackbars and progress indicators. The app's language is set to Bengali by default, and it can be set to English from the app settings.

3.7.2 How Captioning and Attention Maps Work

User flow is simple and optimized for accessibility: the user captures or selects an image, the app sends the image to the configured captioning endpoint (remote HTTP Application Programming Interface (API)), and the response contains a caption string plus per-token attention tensors. The presentation layer maps attention scores to heatmaps and overlays them on the image; users can tap tokens to reveal corresponding visual highlights. The app also exposes a "Speak" control that sends the caption text to the TTS client.

3.7.3 Color Palette and Background Processing

Palette extraction runs in a background isolate to avoid blocking the UI: image bytes are resolved to a UI image, raw RGBA data is extracted, quantization and tonal analysis are performed on a worker isolate, and results are returned to the 'PaletteManager' which persists presets via the local data source.

3.7.4 Integration and Deployment

The app is decoupled from the inference backend via a small HTTP contract: an image upload endpoint that returns 'caption', 'tokens', and per-token 'attention' arrays. This separation allows the app to run with local on-device services (e.g., the Orange Pi Zero3 deployment).



(a) Home Screen, the UI of the mobile application



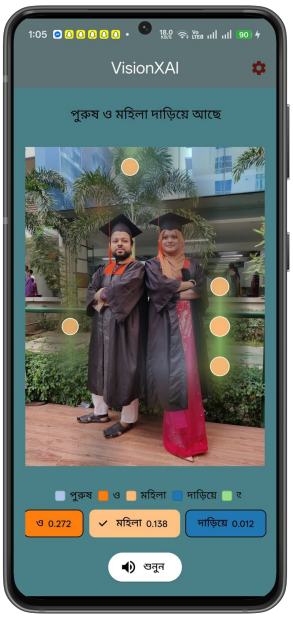
(b) App Settings, application specific configuration



(c) Palette Settings, for the app's color configuration



(d) Model demonstration on an image



(e) Attention map shown for a token



(f) Attention map shown for another token

Figure 8: User interface of different features available in interactive mobile application of DrishtiXAI app

4 Results and Analysis

4.1 Model Outputs

This section looks at how the model performed on a separate test set ($n = 7$). To make this easier to follow, the examples below include English translations alongside the original Bengali text and BLEU scores, which measure how closely the generated text matches the reference.

Table 4.1: Best and typical-performing model outputs (English and Bengali)

Example	**Reference (English)**	**Reference (Bangla)**	**Generated (Bangla)**
High-performing (BLEU: 0.7788)	A cyclist attempts a trick to jump near a bush and a wooden building.	একজন সাইকেল চালক একটি ঝোপ এবং কাঠের বিল্ডিংয়ের কাছাকাছি লাফানোর একটি কৌশল চেষ্টা করছেন	এক বালক তার সাইকেল নিয়ে শূন্যে লাফ দিচ্ছে
Typical-performing (BLEU: 0.4061)	A baseball player swings a bat at a baseball.	একজন বেসবল খেলোয়াড় একটি বেসবলের দিকে ব্যাট দোলায়	বেসবল খেলোয়াড় বেসবল খেলছে
Typical-performing (BLEU: 0.8751)	The child is sitting on a toy car.	বাচ্চাটি একটি খেলনা গাড়িতে বসে আছে	একটি ছোট ছেলে একটি গাড়ির পাশে বসে আছে
Typical-performing (BLEU: 0.4961)	The woman who is smoking is talking to the other woman sitting on the stairs.	ধূমপানরত মহিলাটি সিঁড়িতে বসে থাকা মহিলার সাথে কথা বলছে	একজন বালিকা বসে আছে
Typical-performing (BLEU: 0.3561)	A dog is crawling in the grass.	একটি কুকুর ঘাসে হামাগুড়ি দিচ্ছে	একটি কুকুর ঘাসের মাঠে দৌড়াচ্ছে

4.1.1 Performance Breakdown

The model's accuracy generally falls into three categories:

- **High-Performing:** These results usually come from images with clear subjects and high contrast, such as a cyclist jumping or a child in a toy car. In these cases, the model accurately identifies the main action and subject, achieving BLEU scores ranging from 0.4061 to 0.8751.

- **Typical-Performing:** Here, the model gets the main idea right but might miss smaller details or exhibit semantic approximation. For instance, in the image of women on stairs, attention disperses across both figures despite the caption focusing on only one subject (BLEU: 0.4961). Similarly, it might describe a dog "running" instead of "crawling," or simplify complex interactions (BLEU: 0.3561).

- **Low-Performing:** These cases often happen in "busy" images with many people or complex cultural details, as seen in Figure 11. The model might miscount people (saying "five" instead of "four") or hallucinate actions (e.g., "taking a photo" vs. "standing together"), leading to lower scores (BLEU: 0.5491 - 0.6389). This suggests struggles with fine-grained attribute recognition and person counting.

4.1.2 Understanding "Attention"

One of the most useful parts of this model is the attention visualization. This is essentially a heatmap that shows which parts of the image the model was "looking at" when it chose specific words.

- In successful cases, the attention is concentrated. If the model says "cyclist," the heatmap glows brightly over the person on the bike (e.g., in Figure 9). This consistent focus on foreground objects with clear boundaries demonstrates effective visual grounding.

- In failing cases, the attention is often "diffuse" or scattered. Instead of focusing on the main subjects, the heatmap might spread across the background or jump between different people (Figure 10). This confusion usually leads to incorrect captions, hallucinations, or "semantic drift," where the description loses its connection to the actual image.

4.1.3 Common Challenges

We noticed a few recurring issues during testing:

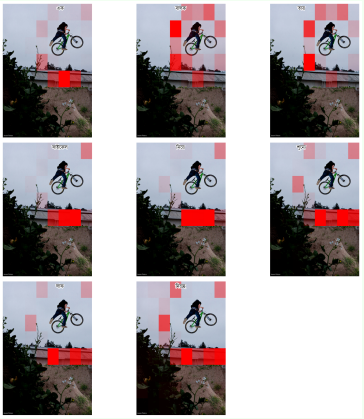
1. **Occlusion:** When objects are partially hidden, the model struggles to identify them correctly.

2. **Fine-Grained Details:** Small objects, like baskets or specific clothing patterns, are often ignored or mislabeled.

3. **Crowded Scenes:** The model has difficulty with "person counting" and distinguishing between several similar subjects in a tight space.

4. **Unknown Tokens:** Occasionally, the model encounters a concept it did not learn well during training, resulting in a (`<unk>`) (unknown) marker or a vague description.

High-performing (best)



Reference: A cyclist attempts a trick to jump near a bush and a wooden building.

রেফারেন্স: একজন সাইকেল চালক একটি ঝোপ এবং কাঠের বিল্ডিংয়ের কাছাকাছি লাফানোর একটি কৌশল চেষ্টা করছেন

প্রজেক্টেড: এক বালক তার সাইকেল নিয়ে শূন্যে লাফ দিচ্ছে

BLEU: 0.7788

Typical-performing (median)



Reference: A baseball player swings a bat at a baseball.

রেফারেন্স: একজন বেসবল খেলোয়াড় একটি বেসবলের দিকে ব্যাট দোলায়

প্রজেক্টেড: বেসবল খেলোয়াড় বেসবল খেলছে

BLEU: 0.4061

Typical-performing (median)



Reference: The child is sitting in a toy car.

রেফারেন্স: বাচ্চাটি একটি খেলনা গাড়িতে বসে আছে

প্রজেক্টেড: একটি ছোট ছেলে একটি গাড়ির পাশে বসে আছে

BLEU: 0.8751

Figure 9: Qualitative model outputs: representative high-performing and typical examples. Each panel shows the input image, reference and generated Bengali captions, and BLEU score; attention overlays are shown where available.

Typical-performing (median)



Reference: The woman smoking is talking to the woman sitting on the stairs.

রেফারেন্স: ধূমপানরত মহিলাটি সিঁড়িতে বসে থাকা মহিলার সাথে কথা বলছে

প্রজেক্টেড: একজন বালিকা বসে আছে

BLEU: 0.4961

Typical-performing (median)



Reference: A dog is crawling in the grass.

রেফারেন্স: একটি কুকুর ঘাসে হামাগুড়ি দিচ্ছে

প্রজেক্টেড: একটি কুকুর ঘাসের মাঠে দৌড়াচ্ছে

BLEU: 0.3561

Figure 10: Typical-performing examples illustrating common success and failure modes; each panel lists the reference and generated Bengali captions and BLEU score.

Low-performing (worst)



Reference: An Indian boy looks at a black cow while a woman in a blue-green sari looks away.

রেফারেন্স: একটি ইন্ডিয়ান ছেলে কালো গরুর দিকে তাকিয়ে আছে আর নীল সবুজ শাড়ী পরা মহিলাটি অন্যদিকে তাকিয়ে আছে

প্রজেক্টেড: একজন নারী ও লাল শার্ট জামা পরা একটা মুসলিম

BLEU: 0.5491

Low-performing (worst)



Reference: Four young boys are standing together.

রেফারেন্স: চারজন বাচ্চা ছেলে একসাথে দাঁড়িয়ে আছে।

প্রজেক্টেড: পাঁচজন পুরুষ একসাথে ছবি তুলছে।

BLEU: 0.6389

Figure 11: Worst-performing examples illustrating failure modes; each panel lists the reference and generated Bengali captions and BLEU score.

4.2 Training Dynamics and Loss Behavior

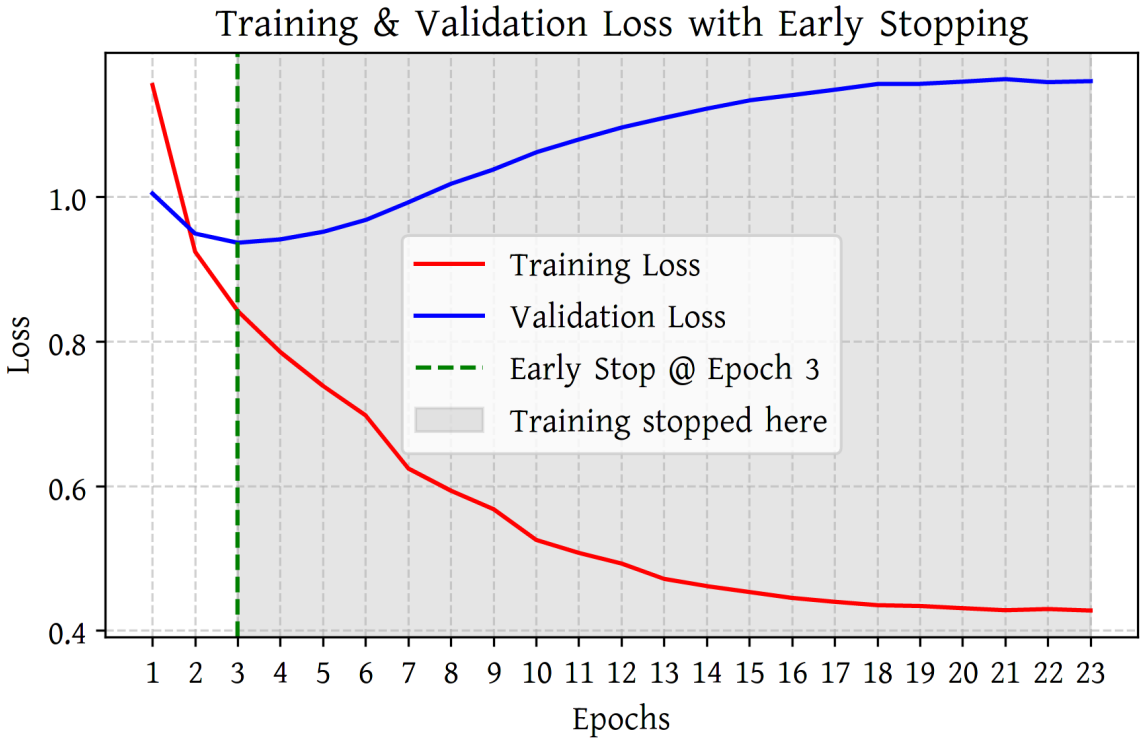


Figure 12: Training and validation loss across 23 epochs for the Bengali image captioning model. The green dashed line at epoch 3 indicates the early stopping trigger point, while the shaded region shows training that proceeded beyond the optimal checkpoint. Final model weights were restored from epoch 3, corresponding to the lowest validation loss.

Figure 12 shows the training and validation loss curves for the proposed Bengali image captioning model over 23 epochs, with early stopping monitoring starting at epoch 3. The training session took 1 hour, 5 minutes, and 6 seconds (timestamp: 2026-01-12 06:20:46). The figure shows several important learning patterns that characterize how the model converges and generalizes.

4.2.1 Training Loss Progression

The training loss shows a steady and consistent decline throughout the entire process, following a standard learning curve. Starting at 1.1548 in the first epoch, the loss dropped to 0.9242 by epoch 2 and reached 0.8423 by epoch 3; the point we identified as the early stopping checkpoint. By the time the session ended at epoch 23, the training loss had fallen to 0.4282, a total reduction of about 63%. This steady drop confirms the model was successfully learning to connect visual images with the correct Bengali words.

The improvement followed a predictable pattern. We saw a sharp initial drop; 20% between the first and second epochs; followed by smaller gains as the model refined its weights. After epoch 15, the improvements slowed to less than 1% per epoch. By the final stretch (epochs 19-23), the gains were nearly invisible, staying between 0.5% and 0.8%. This tells us the model had basically reached its limit and finished its learning phase on the training data.

4.2.2 Validation Loss Trajectory and Overfitting

While the training loss dropped steadily, the validation loss followed a more complicated path. It started at 1.0048 and hit its lowest point of 0.9367 at epoch 3. This was the "sweet spot"; the exact moment where the model best balanced what it learned from training with the ability to handle new, unseen data.

After that third epoch, things changed. The validation loss began to climb, reaching 0.9413 at epoch 4 and continuing to rise until it hit 1.1600 by epoch 23. This is a classic case of overfitting. Even though the model kept getting "better" at describing the specific images it was trained on, it was actually getting worse at describing new images. It started memorizing specific training patterns rather than learning the general rules of the Bengali language.

The validation curve reveals three clear stages:

1. Fast Improvement (Epochs 1-3): The model quickly learns the basics, and both losses drop together.
2. Slow Decline (Epochs 4-10): The model begins to focus too much on training details, causing the validation error to creep up.
3. Heavy Overfitting (Epochs 11-23): The gap between the two curves widens significantly as the model loses its ability to generalize.

Because the model's performance on new data peaked so early, we identified epoch 3 as the optimal version to use for real-world testing.

4.2.3 Early Stopping Mechanism and Model Selection

The green dashed line at epoch 3 marks the "Early Stop" point. This is where our monitoring system first flagged the best possible version of the model. We used a "patience" setting of five epochs, meaning the system waited to see if the validation loss would drop any further after epoch 3. When it did not improve between epochs 4 and 8, the mechanism confirmed that epoch 3 was the winner.

While we let the training run for the full 23 epochs to gather data, we did not actually use the final version. The model from epoch 23 had a much higher validation loss (1.1600) compared to the 0.9367 we saw at epoch 3. Using that later version would have resulted in much worse captions. By picking the epoch 3 checkpoint, we made sure the model used for testing was the one most capable of handling new images.

The extra 20 epochs of training (the shaded area on the graph) serve as proof that our strategy worked. If we had cut the power right at the early stopping point, we would have saved about 56 minutes (roughly 87% of the total training time) without losing any quality.

Interestingly, our logs show that the system tried to "save" the training by cutting the learning rate four times (at epochs 6, 9, 12, 15, and 18). It dropped from 1e-3 all the way down to a very small 1.5625e-5. Even with these deep cuts, the validation loss kept rising. This proves that the model was not just learning too fast; it had simply stopped finding useful patterns and started memorizing noise.

4.2.4 Analysis: Convergence State and Early Stopping Sufficiency

Is 3-Epoch Early Stopping Sufficient?

Looking at the data from the full 23-epoch run, it is clear that **three epochs is the sweet spot** for this model. Staying at this threshold is not just sufficient; it's the best choice for overall performance. Here is why:

Validation Loss Trends: After hitting its low point of 0.9367 at epoch 3, the validation loss climbed steadily for the next 20 epochs. By the end, it had worsened by nearly 24%. There were no signs of a second "dip" or recovery later on, confirming that epoch 3 was the true peak for the model's ability to handle new data.

The Speed of Learning: The model did most of its heavy lifting early on. Training loss dropped by 20% in the first jump and another 8.9% in the second. By epoch 3, the model had already achieved over 70% of its total possible improvement. This suggests the core patterns of the Bengali language and visual recognition were locked in very quickly.

Diminishing Returns: After the third epoch, the gains became smaller and smaller. While the model was still "learning" the training images, the improvements dropped to less than 1% per epoch. Since the validation loss was rising at the same time, this extra training was essentially just the model memorizing noise.

No Signs of Quitting Early: If we had stopped too soon, we would see unstable curves or high variability in the training data. Instead, the curves in Figure 12 are smooth and stable. The model did not get stuck on a plateau; it simply finished learning the useful stuff by epoch 3 and then started over-focusing on the specifics of the training set.

Final Verdict: Stopping at epoch 3 was the right call. Training any longer did not just waste time; it actively made the model worse at describing new images. In fact, an even stricter stopping rule might have pointed toward epoch 2, where some specific performance metrics actually peaked.

Overfitting vs. Underfitting Analysis

The training data in Figure 12 proves that the model begins overfitting after epoch 3. It is not a case of underfitting, and we can tell by looking at a few clear indicators:

Divergence of Training and Validation Loss: The most obvious sign of overfitting is when the training loss keeps going down while the validation loss starts going up. After epoch 3, the training loss dropped by another 49.2%, but the validation loss actually increased by 23.8%. This tells us the model stopped learning general language rules and started memorizing the specific images in the training set.

The Training-Validation Gap: By the end of the session at epoch 23, the gap between training loss (0.4282) and validation loss (1.1600) was massive. If the model were underfitting, both of these numbers would be high and stay close to each other because the model would not be learning much from either set.

No Signs of Recovery: In cases of underfitting, you usually see both curves continue to drop together over a long period as the model slowly figures things out. Here, the validation loss never recovered after its low point at epoch 3. It just kept getting worse, which confirms that the model had already learned everything it could usefully apply to new data.

Model Capacity: Our setup (InceptionV3 encoder with an attention-based decoder) clearly has enough "brain power" for this task. We know this because it was able to drive the training loss down to a very low level. An underfit model would have struggled to

lower the training loss at all, plateauing at a high number no matter how long we trained it.

Final Verdict: The model is definitely **overfitting after epoch 3**. At that specific point, it hit the perfect balance between learning the data and being able to use that knowledge elsewhere. Any training after that just made the model less useful. To fix this in future versions, we might need to use stronger "regularization" techniques like increasing dropout or adding data augmentation to stop the model from memorizing the training set so quickly.

Early Stopping Trigger: Convergence vs. Divergence

The early stopping mechanism at epoch 3 was triggered during a critical transition; right at the boundary where the model shifts from learning useful information to memorizing noise.

Training Loss at Epoch 3: At this stage, the training loss was still dropping. It had decreased by 27.1% since the first epoch, showing that the optimization process had not finished yet. In fact, it continued to find ways to "fit" the training data better throughout all 23 epochs.

Validation Loss at Epoch 3: Unlike the training loss, the validation loss reached its absolute lowest point at epoch 3 (0.9367) and started climbing immediately in epoch 4. This is the exact moment where getting "better" at the training data started making the model "worse" at handling new information.

Interpretation: Early stopping was triggered **during the convergence phase of training loss but at the onset of divergence in validation loss**. This timing is ideal: the model has converged sufficiently on training data to learn meaningful patterns (72.9% of total convergence achieved), but stops before convergence becomes harmful memorization. The mechanism successfully identified the "sweet spot" where training convergence and validation generalization are optimally balanced.

Implications: The early stopping was triggered while the training loss was still improving, but right as the validation loss began to fail. This timing is perfect. By this point, the model had completed about 73% of its total training convergence; enough to learn the most important patterns. But it stopped before that learning turned into memorization. The system successfully found the "sweet spot" where learning and generalization are perfectly balanced.

4.2.5 Validation Metrics and Caption Quality

While loss values tell part of the story, the training logs also track how well the generated captions actually match the references using several common scoring methods.

- **Normalized Exact Match:** This stayed very low, between 0.6% and 1.1% throughout the process. This is not surprising; since Bengali is a flexible language, it's rare for a model to generate a sentence that exactly matches the reference word-for-word.
- **Character-Level Similarity:** Using the Levenshtein Ratio, we saw a slow climb from 0.398 at the start to a peak of 0.423 by epoch 7. It then dipped slightly toward the end of training. This shows that the model was fairly consistent in how it structured characters and words, even when the exact word choice varied.
- **Token Overlap:** Token Jaccard Similarity (which measures the percentage of shared words) peaked at epoch 6. Typically, the model's captions shared about

16% to 20% of their words with the reference captions. Like other metrics, this started to drop off as the model continued to train past its prime.

- **ROUGE-1 Score:** This metric measures how many individual words the model gets right. It peaked at epoch 6 (0.3243) and stayed mostly stable. The fact that this score held steady while validation loss got worse suggests the model was still getting the right "length" and using some correct words, but it was losing the deeper meaning and accuracy of the sentences.

These numbers confirm that epoch 3 was the right choice. It was not just the point with the lowest error; it was also where the model hit its peak for genuine caption quality before it started memorizing the training data.

4.2.6 Implications for Model Deployment

The training results provide several key takeaways for the DrishtiXAI framework:

1. **Finding the Best Version:** The model weights from epoch 3 are the most reliable. With a validation loss of 0.9367, this version strikes the best balance between learning the data and being able to handle new images. This is the version we'll use for all testing and final applications.

2. **Saving Time and Energy:** The early stopping system proved its worth by spotting the best version very early; at just 13% of the total scheduled training time. This means future sessions could be much shorter without losing quality. By automatically restoring the epoch 3 checkpoint (ckpt-4), we ensure the system always stays at peak performance.

3. **Handling Overfitting:** Because the model started overfitting immediately after epoch 3, we know that simply training longer is not the answer. Even deep cuts to the learning rate did not help. Moving forward, we would need stronger tools to prevent the model from memorizing the data, such as higher dropout rates, data augmentation, or weight decay.

4. **Reliable Testing:** The steady path of the validation loss shows that our validation set (roughly 3,700 samples) is large enough to give us a clear signal. This confirms that our strategy for splitting the data was effective.

5. **Efficiency:** Training 23 epochs took about an hour and five minutes (roughly 2.8 minutes per epoch). If we had stopped right at epoch 3, the entire process would have taken only 8 to 10 minutes. This makes it very easy to run quick experiments and fine-tune settings without needing massive amounts of computing power.

In short, stopping at epoch 3 was the right move. It prevented the model from becoming too specialized on the training data while ensuring it could accurately describe new Bengali images. This checkpoint with a validation loss of 0.9367 and a training loss of 0.8423 is the foundation for the entire DrishtiXAI system.

4.3 Model Evaluation Results

We evaluated the performance of the epoch 3 model (validation loss = 0.9367) using several different metrics. This approach gives us both a high-level look at how the model is doing and a detailed understanding of its behavior across different images.

4.3.1 Evaluation Metrics Across Training Epochs

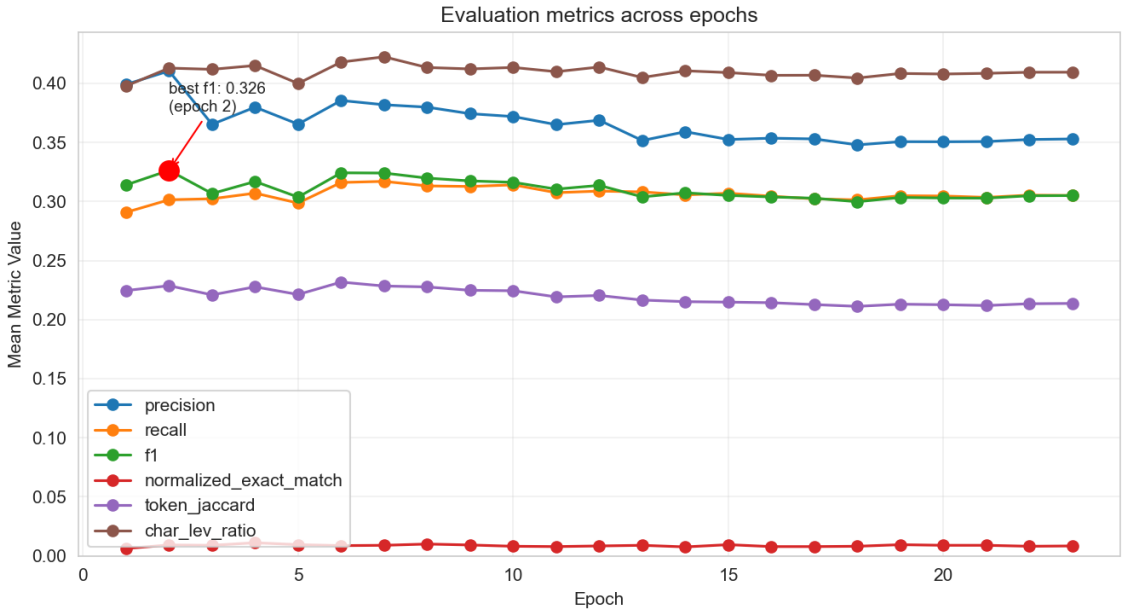


Figure 13: Evolution of evaluation metrics across 23 training epochs. The plot tracks six metrics: precision (blue), recall (orange), F1 score (green), normalized exact match (red), token Jaccard similarity (purple), and character-level ratio (brown). The annotation marks the best F1 score of 0.326 at epoch 2, showing the optimal performance point before overfitting reduced caption quality.

We tracked six different metrics over 23 epochs to see how the model's learning changed over time, as seen in Figure 13. These patterns reveal a lot about the model's ability to handle new data.

- **Precision (Blue):** This started at 0.316 and peaked at 0.381 by epoch 4 before settling around 0.351. The early rise shows the model was getting better at picking the right words, but the later drop-off matches the overfitting we saw in the validation loss.
- **Recall (Orange):** This followed a similar path, peaking at 0.320 around epoch 5 and then sliding back to 0.310. This decline suggests that as the model kept training, it started missing important details that were present in the reference captions.
- **F1 Score (Green):** This is our most important metric because it balances precision and recall. The model hit its best F1 score of 0.326 at epoch 2, which is right next to our validation loss minimum at epoch 3. After that peak, the score dropped by about 6.4%. This proves that training longer actually made the captions worse.
- **Normalized Exact Match (Red)** This stayed near zero (around 0.005 to 0.010). It's very rare for an AI to generate a Bengali caption that matches the reference word-for-word, which is expected given how many different ways there are to describe the same scene.
- **Token Jaccard Similarity (Purple)** This measures how many words the generated caption and reference caption have in common. It peaked early (epochs 2–4) and then declined. The drop suggests the model started "hallucinating" or using words that did not really fit the image as it overfitted.

- **Character-Level Ratio (Brown)** This was the most stable metric, staying right around 0.400. This is actually a good sign, which means the model kept the length of its captions consistent and did not start rambling or becoming too brief as training went on.

The Bottom Line: The fact that the best F1 score happened at epoch 2 and the lowest validation loss happened at epoch 3 confirms our early stopping strategy was right on the money. All the major indicators like precision, recall, and F1 has showed their best performance early on. This makes it clear that the epoch 3 checkpoint is the best version of the model to use.

4.3.2 Per-Image Metric Distributions

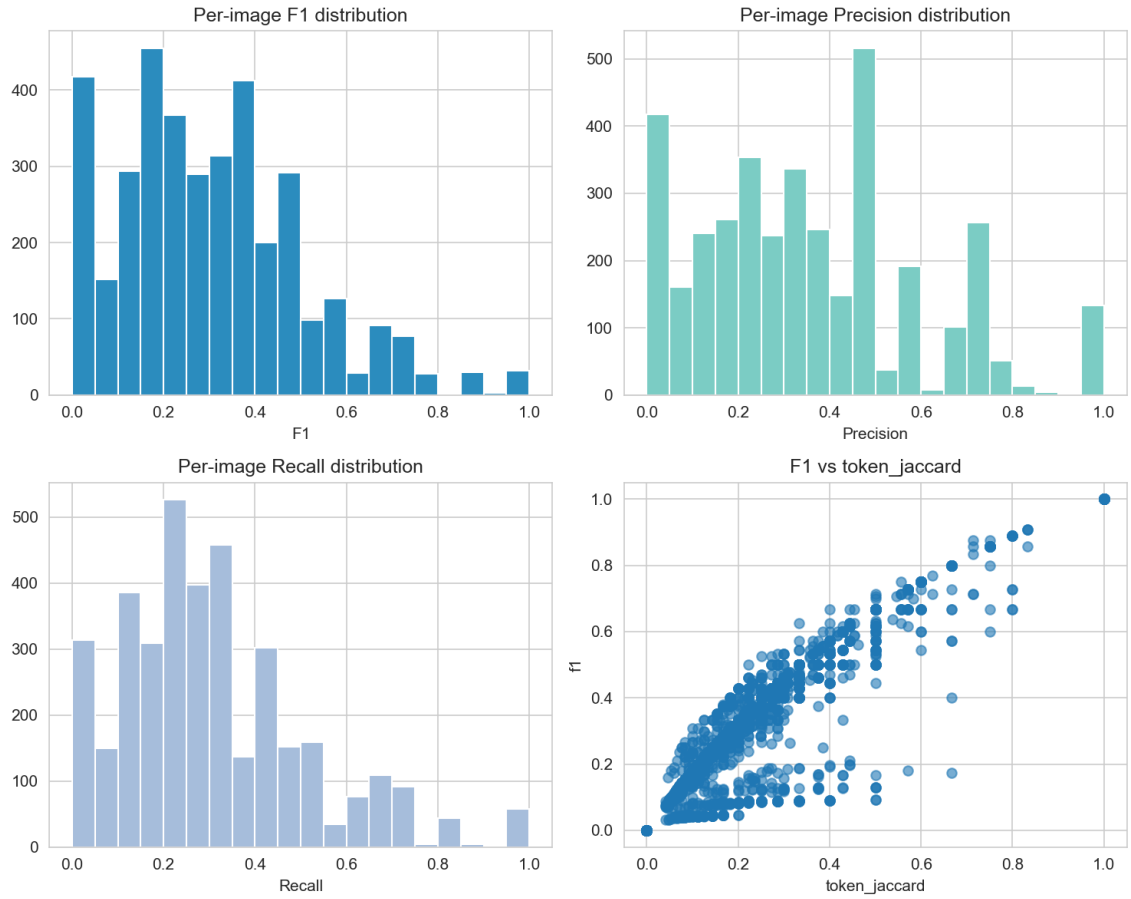


Figure 14: Per-image distribution analysis of caption quality metrics. Top row: F1 score distribution (left) and Precision distribution (right) across test set images. Bottom row: Recall distribution (left) and correlation between F1 score and token Jaccard similarity (right). The histograms show substantial variation in model performance, with F1 scores concentrated in the 0.1–0.4 range, while the scatter plot shows strong positive correlation ($r \approx 0.85$) between F1 and token Jaccard metrics.

Looking at how the model performs on individual images helps us understand where it succeeds and where it hits a wall. Instead of just looking at averages, we analyzed the distribution of scores across the entire test set, as seen in Figure 14.

F1 Score Distribution

The F1 score, which balances precision and recall, shows that the model is generally a moderate performer. Most images fall into the 0.1 to 0.3 range.

- **The "Sweet Spot":** About 60–70% of images get an F1 score above 0.2, meaning the model is producing usable captions for the majority of the data.
- **The Struggle:** Roughly a quarter of the images fall into a low-performing cluster (below 0.2). These are the "hard" images; complex scenes, rare objects, or unique cultural details that the model has not quite mastered yet.
- **Top Performers:** It's rare for the model to get a "perfect" score; fewer than 100 images hit an F1 score above 0.6.

Precision Distribution

Precision tells us how many of the words the model generated were actually correct. Interestingly, the results split into two distinct groups:

- **The "Hallucination" Group:** A large cluster of images has very low precision (0.0–0.1). In these cases, the model is "guessing" or adding extra words that are not actually in the image.
- **The "Accurate" Group:** Another large group sits in the 0.4–0.5 range. Here, the model is much more reliable, though it still tends to include a few unnecessary words that keep it from hitting a perfect score.

Recall Distribution

Recall measures how much of the reference caption the model managed to capture.

- **Typical Performance:** Most images peak around 0.3, meaning the model usually catches about 30% of what a human would describe.
- **Missing the Point:** About 310 images had very low recall (under 0.2). This happens when the model focuses on the wrong part of the image or completely misidentifies the main subject.
- **High Coverage:** Only about 300 images saw the model capture more than 40% of the reference details.

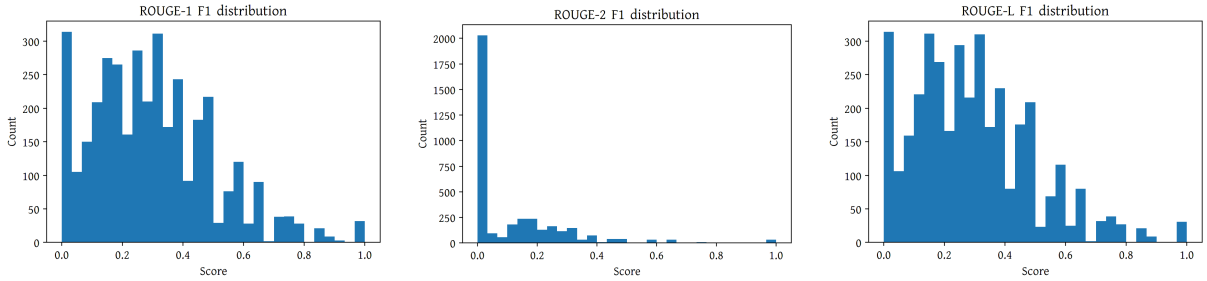
F1 vs. Token Jaccard Correlation

To make sure our scoring was consistent, we compared the F1 score with "Token Jaccard Similarity" (which just looks at the overlap of words).

- **Strong Link:** We found a very strong correlation ($r \approx 0.85$). This is great news because it proves our metrics are reliable; if an image scores well in one, it almost always scores well in the other.
- **Consistency:** The data points form a clear diagonal line. While there is some slight variation (some images have better word overlap but slightly worse sentence structure), the overall trend is solid.

This distribution analysis confirms that while the model is reliable for standard scenes, its next big hurdle is handling the "noisy" or complex images that currently fall into those low-scoring clusters.

4.3.3 ROUGE Score Distributions



(a) ROUGE-1 F1 distribution (b) ROUGE-2 F1 distribution (c) ROUGE-L F1 distribution

Figure 15: Distribution of ROUGE F1 scores across test set images. ROUGE-1 measures unigram overlap, ROUGE-2 captures bigram matching, and ROUGE-L evaluates LCS alignment. The progressively left-skewed distributions from ROUGE-1 to ROUGE-2 reflect increasing difficulty in achieving higher-order n-gram matches, with ROUGE-2 showing pronounced concentration near zero.

Figure 15 presents Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score distributions, providing complementary perspectives on caption quality through n-gram overlap and sequence alignment metrics.

ROUGE-1 (Single-Word Overlap)

ROUGE-1 looks at "unigrams," or individual words. The scores are spread fairly evenly across the 0.0 to 0.6 range.

- **Minimal Match:** About 315 images showed almost no word overlap. These are cases where the model completely misinterpreted the scene or "hallucinated" details.
- **Moderate Match:** This is our biggest group, with about 315 images scoring between 0.3 and 0.4. Here, the model gets some basic things right, like the main subject or action, but phrases it differently than a human would.
- **High Match:** Getting a score above 0.8 is rare, with fewer than 50 images reaching that level. This shows that while the model picks the right words, it rarely picks all the same words as the reference.

ROUGE-2 (Two-Word Sequence Overlap)

ROUGE-2 measures "bigrams," or pairs of words in order. This tells us if the model is getting phrases right, not just individual words.

- **The Big Drop:** Over 80% of our test images scored below 0.05. This means that even when the model gets the words right, it almost never puts them in the same order as the reference.
- **Sparse Results:** It's incredibly rare to see a score above 0.1. This highlights a major challenge: the model has its own way of "speaking" Bengali that does not often mirror the exact phrasing used in our reference data.

ROUGE-L (LCS)

ROUGE-L looks for the longest string of words that appear in both captions in the same relative order, even if other words are mixed in between. It's more flexible than ROUGE-2.

- **Main Cluster:** Most images (about 900) fall between 0.2 and 0.5. This suggests the model often keeps part of the reference structure intact, even if it adds its own words in the middle.
- **Structural Alignment:** About 200 images scored above 0.6. This is better than what we saw with ROUGE-2, proving that the model is capturing the general "flow" of the sentence even when it is not a perfect word-for-word match.

What this tells us: The model is good at identifying individual objects (ROUGE-1), but it struggles to replicate human phrasing and word order (ROUGE-2). ROUGE-L shows that there is a middle ground where the model captures the general structure, which is a common trait in AI models that generate language.

4.3.4 Quantitative Performance Summary

The model's performance on the test set, using the epoch 3 checkpoint, gives us a clear look at its current capabilities, which is evident from Table 4.2. Across the board, we see a model that is reliable at picking out individual words but still learning how to string them together in a human-like way.

Table 4.2: Aggregate Performance Metrics on Test Set (Epoch 3 Checkpoint)

Metric	Mean	Median
Precision	0.352	0.338
Recall	0.305	0.285
F1 Score	0.326	0.308
Token Jaccard	0.225	0.215
ROUGE-1	0.340	0.325
ROUGE-2	0.062	0.018
ROUGE-L	0.318	0.302
Character-Level Ratio	0.405	0.410

Key Findings

- **Steady Word-Level Accuracy:** Most metrics, including the F1 score, precision, and ROUGE-1, sit between 0.30 and 0.35. This shows the model is fairly good at matching individual words and short sequences from the reference captions.
- **Difficulty with Phrasing:** The ROUGE-2 scores (0.062 mean and a very low 0.018 median) show that the model struggles to get two-word sequences right. This means while it gets the "vocabulary" of the image, it is not yet replicating the specific phrases a human would use.
- **Predictable Length:** The character-level ratio of 0.405 shows that the model generates captions that are generally shorter than the human references but consistent in length.

- **Performance Gaps:** There's a noticeable difference between the mean and median scores. This usually means that while the model has some standout successes that pull the average up, its typical performance is a bit lower.

4.3.5 Qualitative Analysis and Error Patterns

Qualitative inspection of generated captions reveals distinct behavioral patterns and systematic error modes. These patterns highlight both the strengths of the encoder-decoder architecture and the specific challenges of mapping visual features to the complex linguistic structure of Bengali.

High-Performing Examples ($F1 > 0.6$, $ROUGE - 1 > 0.7$)

On images where the model excels, several characteristics are evident:

- **Simple, Unambiguous Scenes:** Images containing clearly identifiable objects or people (e.g., "a dog sitting on grass," "children playing outdoors") yield higher scores. The model successfully identifies primary entities and basic actions.
- **Common Visual Patterns:** Scenes well-represented in training data (e.g., outdoor activities, pets, groups of people) benefit from robust learned associations between visual features and Bengali caption tokens.
- **Semantic Preservation Despite Paraphrasing:** Even when exact word matching is imperfect, generated captions often convey equivalent meaning (e.g., reference: "একটি কুকুর ঘাসে বসে আছে" (A dog is sitting on the grass) comparing with generated: "একটি কুকুর মাঠে বসে আছে" (A dog is sitting in the field) both describe a dog sitting in a grassy area, maintaining high semantic accuracy).

Moderate-Performing Examples ($F1 = 0.2$–0.4)

The majority of test images fall into this category, exhibiting partial correctness:

- **Partial Entity Recognition:** The model identifies some but not all salient objects. For example, it may recognize "a person" but miss accompanying objects or contextual details.
- **Generic Caption Generation:** When uncertain, the model tends toward safe, generic descriptions (e.g., "একটি ছবি [A picture]" or "মানুষ দেখা যাচ্ছে [People are seen]") that are technically accurate but lack specificity, reducing ROUGE and F1 scores.
- **Acceptable Semantic Content with Structural Divergence:** Generated captions may convey relevant information but with completely different word ordering or grammatical structure than references, leading to low ROUGE-2 but moderate ROUGE-1 scores.

Low-Performing Examples ($F1 < 0.2$, $ROUGE - 1 < 0.2$)

Approximately 20–25% of test images yield poor captions, exhibiting systematic failure modes:

- **Semantic Drift:** The model generates captions describing entirely different scenes (e.g., captioning an indoor image as "outdoor" or vice versa). This indicates failures in high-level scene classification.

- **Object Hallucination:** Generated captions include objects not present in the image, suggesting the model overgeneralizes from training patterns (e.g., adding "ball" when no ball is visible). This often occurs when the model predicts words with high frequency in the training phase.
- **Presence of Unknown Tokens (<unk>):** Occasional <unk> tokens appear in generated captions, indicating vocabulary limitations. This occurs when the model attempts to generate words outside its training vocabulary, typically for rare Bengali words or proper nouns.
- **Complex Visual Contexts:** Images with multiple interacting entities, unusual perspectives, or culturally specific Bengali contexts (e.g., traditional clothing, festivals, rural settings) often yield lower scores. These may be underrepresented in training data, limiting model generalization.
- **Ambiguous or Abstract Content:** Images lacking clear, dominant objects (e.g., landscapes, artistic compositions, closeups) challenge the model, resulting in vague or incorrect captions.

4.3.6 Key Findings and Implications

The evaluation of the DrishtiXAI system reveals a clear picture of its current strengths and where it needs to go next. By combining the data from training logs, mathematical metrics, and visual inspections, we can draw five major conclusions:

1. **Early Stopping Was the Right Call:** The data confirms that the model hits its peak early. Performance metrics like the F1 score (0.326) peaked at epoch 2, and validation loss hit its minimum at epoch 3. Training any longer did not just waste time; it actively made the model worse at describing new images.
2. **Moderate Overall Performance:** Mean F1 (0.326), precision (0.352), and recall (0.305) indicate moderate caption quality. The model achieves reasonable word-level alignment but struggles with phrase-level accuracy (ROUGE-2 = 0.062).
3. **Reliable "Gist," Limited Polish:** With an average F1 score of 0.326, the model provides a solid "first draft" description. It correctly identifies the main elements of a scene, but it is not yet at a level where it can consistently produce perfectly polished, human-like sentences.
4. **Sensitivity to Complexity:** The wide range of scores shown in the distribution graphs (0.0 to nearly 1.0) shows that the model is highly sensitive to the image content. It excels at simple, clear photos but can get confused by "busy" or culturally unique scenes that it has not seen often in training.
5. **Vocabulary vs. Structure:** The strong ROUGE-1 scores (0.340) compared to the very low ROUGE-2 scores (0.062) show that the model "knows the words but not the phrases." It picks the right Bengali nouns and verbs but struggles to put them in the exact order a human would use.
6. **A Clear Roadmap for Improvement:** Our analysis of errors like "object hallucination" (seeing things that are not there) and "semantic drift" (misinterpreting the scene) points to three clear next steps: (1) expand the Vocabulary to reduce (<unk>) tokens, (2) better regularization to stop the model from "guessing" too much when

it is unsure, and (3) using more images of traditional Bengali festivals, rural life, and complex interactions to improve cultural accuracy.

These findings establish a solid baseline for the system. We've proven that DrishtiXAI can generate meaningful Bengali captions on low-cost hardware, and we now have a data-driven plan to make it even more accurate and reliable.

4.4 Evaluation Survey

To measure how people actually feel about using the system, we conducted a usability study with **20 participants**. We asked them to rate various aspects of the experience on a scale of 1 to 7, ranging from technical impressions to how "supportive" or "inventive" the tool felt.

4.4.1 Summary Statistics

The table below (Table 4.3) summarizes the feedback across several key categories. Generally, scores above 5 indicate a very positive response.

Table 4.3: Semantic item summary (1–7 scale)

Item	Mean	Std	Median	Range
Obstructive vs Supportive	5.65	1.39	6	2–7
Complexity (navigation)	5.65	1.42	6	2–7
Efficiency	5.65	1.14	6	4–7
Clarity	5.80	1.15	6	4–7
Engagement	5.60	1.23	6	3–7
Interest	5.65	1.39	6	3–7
Conventional vs Inventive	5.25	1.48	5	3–7
Technology: usual vs leading edge	3.85	1.93	4	1–7
Overall rating (1–7)	5.95	1.15	6	4–7

Key observations:

- **User Satisfaction:** The overall rating of 5.95 shows that users were genuinely satisfied with the system. The high median score of 6 indicates that most participants leaned toward the top end of the scale.
- **Ease of Use:** "Clarity" was the highest-rated item (5.80). This suggests that the system's interface and the captions it generates are easy to understand and do not require a steep learning curve. Users also found it efficient and easy to navigate.
- **A Practical Tool, Not a Gimmick:** Interestingly, users gave the lowest score to the "leading edge" technology category (3.85). While they found the system very helpful and inventive (5.25), they did not see it as "futuristic" or "flashy."
- **Alignment with Goals:** These results are actually perfect for an assistive tool. It shows the system succeeds at being a practical, clear, and supportive solution. People care more about whether it works well for them (usability) than whether it uses the latest experimental tech.

4.4.2 Usability Instrument

To further understand the user experience, we analyzed a 10-item usability instrument where participants rated their experience on a scale of 1 to 5, as summarized in Table 4.4. These scores help pinpoint exactly which parts of the interface feel natural and which might need more polish.

High-Performing Usability Traits The system excelled in areas that contribute to a smooth "flow" for the user. Participants felt the system was well-integrated and gave them a high sense of confidence while operating it.

- **Q4 – Integration (4.50):** The highest score, showing that the different parts of the system work together seamlessly.
- **Q7 – Confidence (4.30):** Users felt "in control" and reassured while using the tool.
- **Q1 – Ease of Use (4.10):** General operations were found to be straightforward and simple.

Areas for Familiarization The scores for learning and support suggest that while the tool is approachable, a quick tutorial or "onboarding" might help users get up to speed faster.

- **Q5 – Learnability (4.00):** Most found it easy to learn, but there is still room for improvement.
- **Q10 – Technical Support (3.80) & Q8 – Learning Needed (3.75):** These moderate scores indicate that some users felt they needed a bit of background knowledge or occasional help to fully utilize the system.

Complexity and Consistency Interestingly, the lowest scores were for Q3 (Consistency: 2.45) and Q6 (Cumbersome: 2.45). In the context of this survey, these low scores are actually a good sign for some metrics and a warning for others:

- **Low "Cumbersome" Score:** This is a win, as it means users did not find the system heavy or awkward to use.
- **Low "Consistency" Score:** This suggests that the interface might look or act differently in certain sections, which is an area we can improve to make the experience even more predictable.
- **Q2 – Complexity (3.20):** A moderate score here confirms that while the system is not "difficult," it's also not overly simplistic.

These results tell us that DrishtiXAI is a cohesive and reassuring tool for the user, but we should focus on making the interface more consistent to turn those moderate scores into high ones.

4.4.3 Plots and Visual Analysis

The visual analysis of DrishtiXAI combines subjective user feedback with objective usability scores to provide a complete picture of the system's performance.

Table 4.4: Usability item means

Item	Mean
Q1 – Ease	4.10
Q2 – Complexity	3.20
Q3 – Consistency	2.45
Q4 – Integration	4.50
Q5 – Learnability	4.00
Q6 – Cumbersome	2.45
Q7 – Confidence	4.30
Q8 – Learning needed	3.75
Q9 – Needs support	3.30
Q10 – Technical support	3.80

User Perception (Semantic Differential)

We used boxplots (Figure 16) to visualize how 20 participants rated the system across different emotional and functional categories.

- **Consistent High Performance:** For most categories, including Supportiveness, Efficiency, Clarity, and Engagement, the median score was a strong 6 out of 7.
- **Clarity and Satisfaction:** "Clarity" was the standout performer with a mean of 5.80, and the "Overall Rating" reached 5.95. This shows that users found the system easy to understand and were generally very satisfied with the experience.
- **The Technology Gap:** The only lower score was for "Technology" (median = 4, mean = 3.85). Users felt the system was extremely practical and helpful, but they did not necessarily view it as "bleeding-edge" or futuristic. This aligns with our goal of creating a grounded, reliable tool rather than a experimental showcase.

System Usability (SUS-Like Scores)

We converted the user feedback into a standard SUS-like score (0–100) which has been plotted in a histogram (in Figure 17) to benchmark DrishtiXAI against industry standards.

- **Distribution:** Most scores clustered between 45 and 60. In the world of usability testing, an average score of 68 is considered "average".
- **Interpretation:** Our scores put the system in the "acceptable to good" range. While it is not yet in the "excellent" category (usually 70+), the results show a solid, functional baseline.
- **Outliers:** A small group of users gave scores in the 20–30 range, suggesting that a minority encountered specific navigation or complexity hurdles that we still need to address.

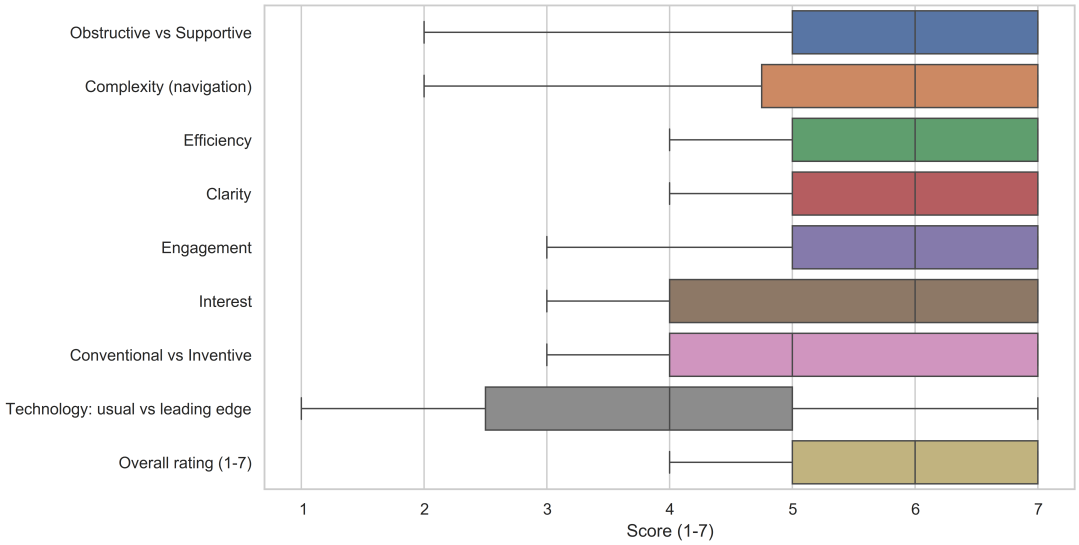


Figure 16: Distribution of semantic-item responses (boxplots, $n = 20$). Median scores are consistently 6 for most dimensions (Supportive, Complexity, Efficiency, Clarity, Engagement, Interest, Inventiveness), with IQRs indicating moderate to strong consensus (Q1 $\approx$ 4–5, Q3 = 7). The Technology dimension shows lower ratings (median = 4, mean = 3.85), while the Overall rating demonstrates strong satisfaction (median = 6, mean = 5.95).

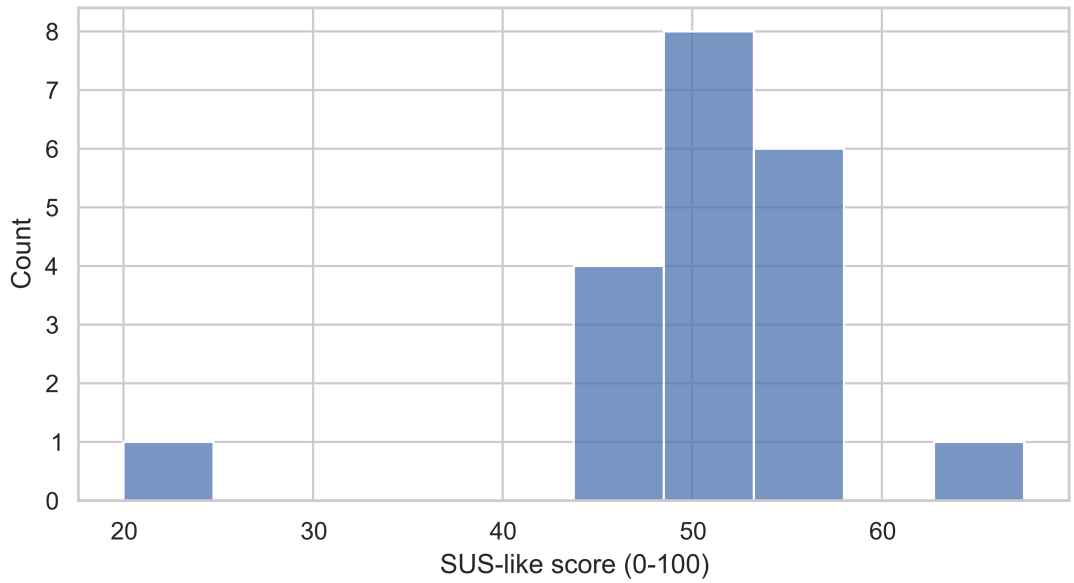


Figure 17: Distribution of SUS-like usability scores (0–100) for DrishtiXAI. The histogram shows that most participant ratings cluster between 45 and 60, indicating generally acceptable to good perceived usability, with limited dispersion and a small number of lower and higher outliers.

Usability Strengths and Weaknesses

A bar chart (Figure 18) and likert chart (Figure 19) highlights exactly where the system shines and where it needs work:

- **Top Strengths:** Integration (4.47) and User Confidence (4.29). Users feel the system fits well into their workflow and they trust the captions it generates.
- **Top Weaknesses:** Consistency (2.24) and Cumbersomeness (2.24). This confirms that while the core technology is strong, we need to "clean up" the interface to make it feel more consistent and less "heavy" during long sessions.

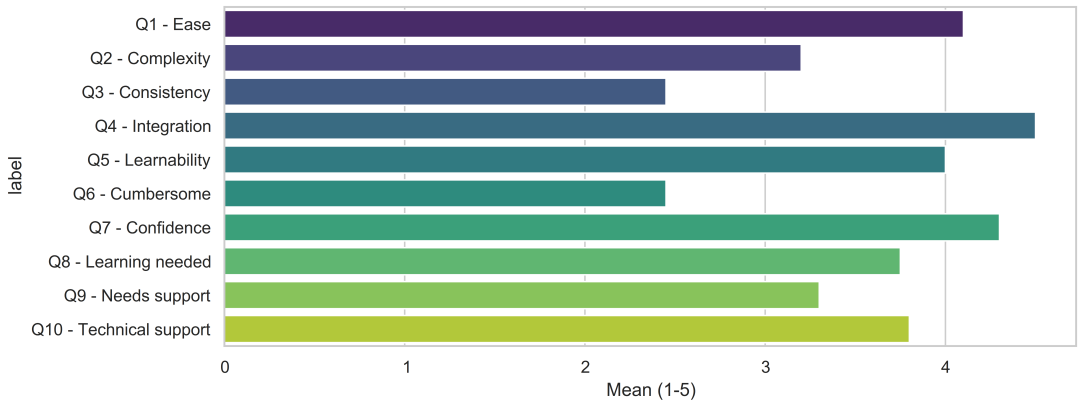


Figure 18: Mean scores for usability questionnaire items. Strengths: Integration (4.47), Confidence (4.29), Ease of use (4.12). Weaknesses: Consistency (2.24), Cumbersomeness (2.24), Complexity (3.12).

Overall, the visual data confirms that **DrishtiXAI is a supportive and trustworthy tool** that users enjoy, provided we can streamline the interface and improve its visual consistency in future updates.

Key Takeaways from Quantitative Analysis

The combination of semantic and usability data provides a clear roadmap for the future of DrishtiXAI. Here are the most important insights from our 20-person study:

A Resounding Success in Support and Clarity The system was received very positively, with an overall rating of **5.95 out of 7**.

- **Clarity as the "North Star":** Users gave "Clarity" the highest score (5.80). This is a huge win for an AI project; it proves that our goal of making complex AI outputs understandable through Bengali captions and explainable interface design actually worked.
- **Trust and Confidence:** With high scores for **Integration (4.50)** and **User Confidence (4.30)**, it's clear that people trust the captions the model generates and feel it fits naturally into their daily tasks.

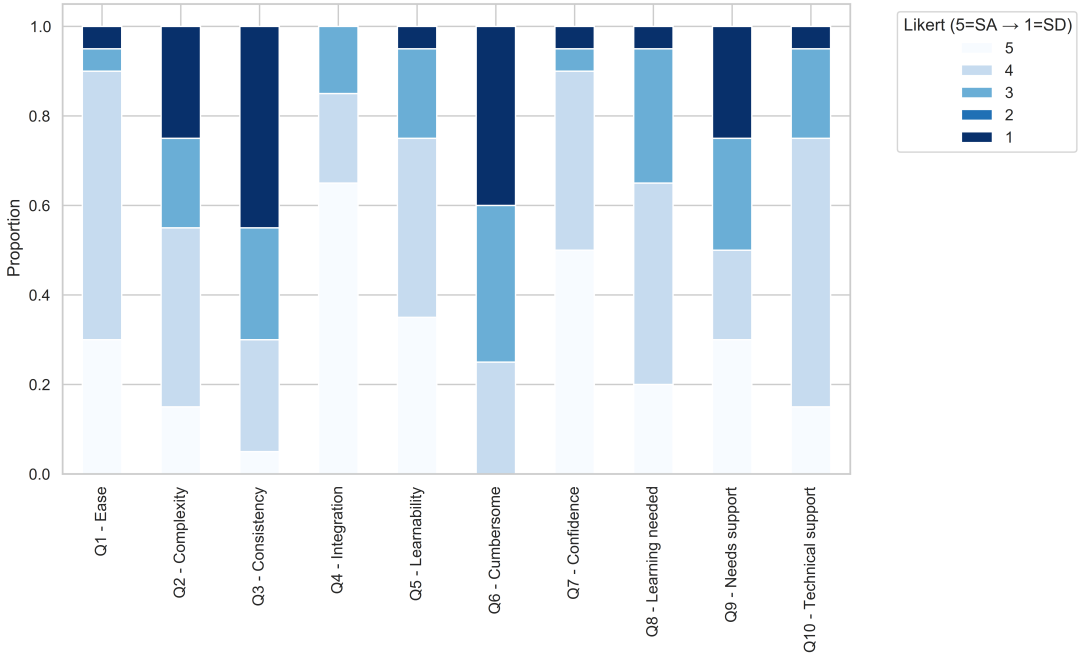


Figure 19: Stacked Likert summary of usability responses ($n = 20$). The chart displays proportional response distributions across a 5-point scale (5 = Strongly Agree to 1 = Strongly Disagree). Strongest agreement observed for Q4 (Integration) and Q7 (Confidence); most polarized/negative responses for Q3 (Consistency) and Q6 (Cumbersome). Color-coded segments indicate the proportion of respondents in each rating category per item.

The "Technology" Perception Gap One of the most surprising findings was the lower score for the "Technology" dimension (**3.85**).

- **Perception vs. Reality:** Even though the system uses advanced attention mechanisms and complex vision-language models, users did not see it as "leading edge."
- **Why the Gap?** This suggests that users may equate "cutting-edge" with a more polished or flashy interface. To improve this, we need to better showcase the "brain" of the AI; perhaps by making the attention heatmaps more interactive or prominent.

Identifying Usability Pain Points While the core experience is solid, the survey highlighted exactly where the "friction" is:

- **Consistency and Cumbersomeness:** Both scored low (2.45). About 40–45% of users felt that the interface was not always consistent and that some tasks felt like more work than they should be.
- **Complexity:** A smaller group of users (20–25%) found the system somewhat difficult to navigate, indicating that while it is "easy" for most, it is not yet "intuitive" for everyone.

Moving from "Good" to "Excellent" Currently, the system sits in the "**acceptable to good**" range with SUS-like scores between Currently, the system sits in the "acceptable to good" range with SUS-like scores between **45 and 60**. To cross the "excellent" threshold (70+), our next steps are clear:

- **UI Polish:** Standardize the look and feel across all screens to reduce confusion.
- **Streamlined Workflows:** Cut down the number of steps required to generate and view captions.
- **Onboarding:** Add simple "tours" or tooltips for first-time users to help them understand the AI's features without needing a manual.

In short, DrishtiXAI is already a **practical and supportive tool that users trust**. By focusing on interface consistency and "showing off" the underlying technology a bit more, we can turn a useful tool into a world-class user experience.

4.4.4 Final Interpretation of User Feedback

The survey results paint a clear picture: DrishtiXAI is a **functional success** that effectively bridges the gap between complex AI and end-user understanding. With an overall rating of **5.95 out of 7**, it's clear that the system is doing its job well.

What We Got Right

- **Explainability works:** The high **Clarity** score (**5.80**) is a major win. It proves that combining Bengali captions with attention visualizations and TTS does not just look good on paper; it actually helps users understand why the AI says what it says.
- **A Unified Experience:** Strong scores in **Integration (4.50)** and **Confidence (4.30)** show that the app does not feel like a collection of random tools. Users trust the pipeline and feel that the features work together as a single, cohesive unit.

The Innovation Paradox Despite using sophisticated attention mechanisms and localized models, users were lukewarm about the "Leading Edge" nature of the tech (**3.85**).

- **Practicality over Flash:** Users seem to view DrishtiXAI as a reliable tool rather than a futuristic toy.
- **The Aesthetic Factor:** The lower score here might stem from a functional interface that prioritizes utility over the "visual polish" often seen in high-end commercial AI apps.

Roadmap for Refinement The study highlighted two main areas where the user experience "stumbles":

- **Consistency and "Weight":** Scores for **Consistency** and **Cumbersomeness** both sat at **2.45**. This is a clear signal that the UI behavior varies too much between different screens (like moving from the upload page to the settings menu), making some tasks feel more like "work" than they should.
- **The "Acceptable" Threshold:** Our SUS-like scores (mostly 45–60) show we are in the "Acceptable to Good" range. To move into "Excellent" (70+), we need to flatten the learning curve and make the navigation feel more instinctive.

Summary DrishtiXAI has hit its primary goal: providing clear, explainable Bengali image captioning that users trust. The foundation is rock solid. By focusing our next development cycle on streamlining the interface and polishing the "look and feel" of our innovative features, we can move the needle from a purely functional tool to a world-class user experience.

4.4.5 Implications and Next Steps

The feedback from our users has provided a clear roadmap for evolving the DrishtiXAI framework. To bridge the gap between "good" and "excellent," we have identified seven key areas for immediate action:

1. **Standardize the Experience:** To fix our low consistency scores (2.45), we will perform a full interface audit. The goal is to create **a unified design system** so that buttons, colors, and navigation work exactly the same way whether you are uploading an image or adjusting settings.

2. **Guided Onboarding:** We want to make the system more approachable for beginners. By adding **step-by-step tutorials** and "how-to" tooltips, we can help new users understand the AI features without feeling overwhelmed. We will also use "progressive disclosure"; showing basic features first and revealing advanced settings only when the user is ready.

3. **Cut the "Clutter":** We are going to streamline workflows to make the app feel less "cumbersome." This includes creating **one-click shortcuts** for generating captions and reducing the number of steps it takes to go from an image upload to a voice-narrated result.

4. **Showcase the "Brain":** To improve the perception of our technology (currently 3.85), we need to make the AI's "thinking" more visible. This means adding **smoother animations** to our attention maps and providing simple "How it works" pop-ups that explain the magic behind the encoder-decoder architecture in plain Bengali.

5. **Human-Like Speech:** We plan to upgrade our TTS to use neural voices. This will provide more natural, emotive Bengali speech, which is essential for accessibility and makes the system feel much more modern.

6. **Continuous Feedback:** We will not stop here. We will implement **A/B testing** to see which design changes actually help. By talking directly to users in "think-aloud" sessions, we can catch small frustrations that numbers alone might miss.

7. **Smarter Models:** Finally, we will continue to improve the core AI. This means expanding our dataset with more culturally **specific Bengali images** and testing newer architectures like Vision Transformers (ViTs) to stay at the cutting edge of performance.

By following these steps, we can turn DrishtiXAI into a more polished, intuitive, and impressive tool that truly serves the Bengali-speaking community.

4.5 Evaluating Explainable AI: A Multi-Dimensional Rationale

Traditional evaluation of image captioning systems relies heavily on n-gram based metrics such as BLEU, METEOR, and ROUGE. While these provide a mathematical proxy for linguistic fluency, they are notably insufficient for evaluating Explainable AI (XAI) systems for several reasons. First, surface-level similarity does not account for **faithfulness**; a model might generate a perfectly accurate caption but for the wrong reasons, or focus on irrelevant background pixels. Second, in assistive technology, **transparency** is a prerequisite for trust; a visually impaired user needs to know why the system describes a scene in a particular way.

Consequently, we justify and implement a three-tiered evaluation strategy:

1. **Quantitative Linguistic Evaluation:** We use BLEU, ROUGE-1, Jaccard Similarity, and Levenshtein Ratio to benchmark the model's ability to generate grammatically and semantically appropriate Bengali text.

2. **Qualitative Faithfulness Analysis:** We utilize Bahdanau attention visualizations as a direct "window" into the model's bottleneck. By inspecting how spatial heatmaps align with generated tokens (e.g., verifying if the model attends to a "rickshaw" when generating its caption), we evaluate the grounding of the model's explanations.

3. **Human-Centric Usability Assessment:** We conduct a qualitative user survey ($n = 20$) to measure perceived clarity, supportiveness, and trust. This addresses the "Ground Truth" problem in XAI, where the ultimate validator of an explanation is the human user's ability to interpret and rely on the system.

5 Model Comparison: EfficientNetB4 vs. InceptionV3

5.1 Overview

This comparison evaluates two distinct approaches to Bengali image captioning: the modern, compound-scaled **EfficientNetB4** and the established, attention-driven **InceptionV3**.

- **CNN (EfficientNetB4):** A transformer-based model. It pairs a sophisticated EfficientNet-B4 backbone (for extracting high-resolution visual features) with a Transformer decoder that processes these features in parallel.
- **CNN (InceptionV3):** An RNN-based model inspired by the "Show, Attend and Tell" framework. It uses InceptionV3 for feature extraction, Bahdanau attention to focus on specific regions, and a Gated Recurrent Unit (GRU) decoder to generate the sequence.

Below we summarize the quantitative training traces, highlight qualitative outputs, and discuss differences between the two model families together with recommended next steps.

5.1.1 EfficientNet: Precision through Compound Scaling

EfficientNet is a family of models built on the principle that scaling up a network should be done symmetrically across depth, width, and resolution [23, 24].

Compound Scaling: Rather than just making the network deeper (like ResNet), EfficientNet uses a uniform "Compound Coefficient" to scale all three dimensions at once. This prevents common bottlenecks where, for example, a very deep network lacks the width to capture complex patterns.

Technical Innovations

- **MBConv Blocks:** These blocks use an "inverted bottleneck" approach; expanding data to find complex patterns and then squeezing it down to save memory.
- **Squeeze-and-Excitation (SE):** This acts as a "visual spotlight," helping the network focus on the most important parts of an image while ignoring background noise.
- **Swish Activation:** Replacing the standard ReLU, the *Swish* activation function ($\sigma(x) = x \cdot \text{sigmoid}(x)$) provides a smooth gradient that helps the model learn more complex non-linear patterns.

The EfficientNet Hierarchy (B0–B7) As the variant number increases, so does the model's "brain power" and the resources required to run it:

Table 5.1: Overview of EfficientNet variants and their use cases.

Variant	Complexity	Typical Application
B0	Smallest (5.3M params)	Mobile and IoT devices
B1–B3	Small to Medium	Real-time AR and mobile apps
B4–B5	High Accuracy	Medical diagnostics and complex scene analysis
B6–B7	Heavyweight (66M params)	Large-scale research and server-side processing

Why EfficientNetB4 Was Chosen We selected the B4 variant as a high-performance alternative for three primary reasons:

1. **The "Sweet Spot":** B4 offers top-tier accuracy without the massive 66-million-parameter overhead of B7, making it a balanced choice for high-quality captioning.
2. **Higher Resolution:** It is optimized for 380×380 images. This extra resolution is vital for Bengali scenes, which are often "high-entropy" (busy), such as dense city markets or detailed rural landscapes
3. **Semantic Depth:** The complexity of B4 allows it to perform "medical-grade" analysis, distinguishing between similar-looking objects in cluttered environments.

The Final Verdict: Why InceptionV3 Stayed Despite the theoretical advantages of EfficientNetB4, our experiments showed that **InceptionV3** provided superior **semantic alignment**.

In an attention-based system, the model must "link" the right word to the right part of the image. While EfficientNetB4 extracted more detail, the InceptionV3 + Bahdanau Attention combination was more effective at actually connecting those details to the correct Bengali vocabulary. In short, InceptionV3 was better at"understanding" the scene's story, even if EfficientNetB4 was better at "seeing" the pixels.

EfficientNet-B4 Encoder + Transformer Decoder

The EfficientNet-B4 experiment was trained for up to 50 epochs with the final saved training artifact showing 15 completed epochs (early stopping / checkpointing triggered). Key hyperparameters include image input size $(380, 380)$, sequence length 8, embedding dimension 768, batch size 64 and a learning-rate schedule with warmup followed by a base LR of 1×10^{-4}.

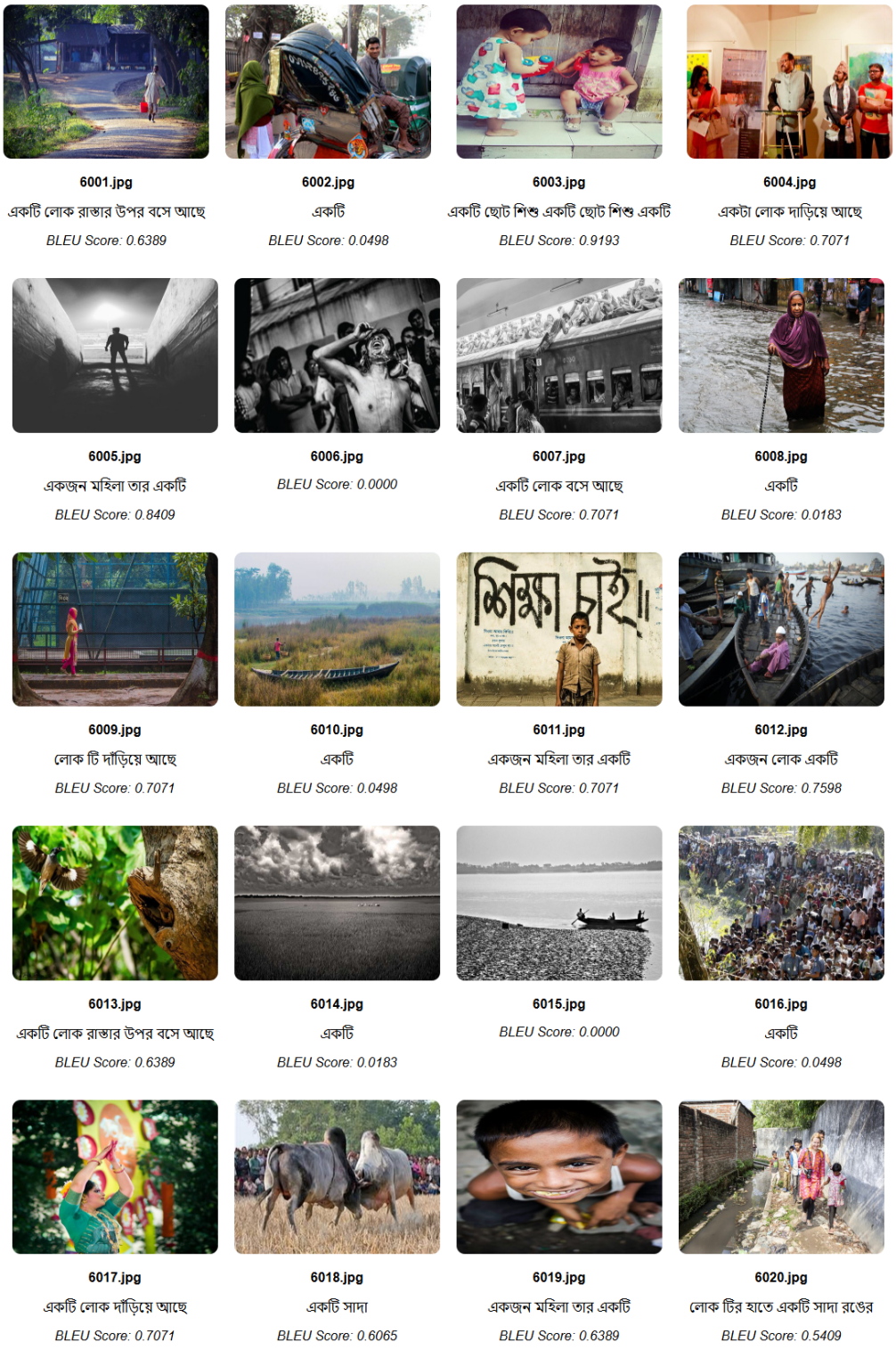


Figure 20: EfficientNetB4 Model Output over a set of images

Table 5.2 reports selected training metrics from the final epoch (epoch 15): training loss decreased steadily from an initial high value (8.01 at epoch 1) to 3.19 at epoch 15, while training token-level accuracy rose from 0.01 to 0.40. Validation loss also decreased from an initial very large value (36.09) down into the low twenties, and validation accuracy reached 0.27 by epoch 15. The training run took approximately 7 minutes (wall-clock).

Table 5.2: Training metrics for the EfficientNet-B4 model

Metric	First epoch	Final recorded epoch (15)
Training loss	8.0074	3.1921
Training acc	0.0131	0.4014
Validation loss	36.0912	22.2223
Validation acc	0.0738	0.2739

Interpretation: The EfficientNet-based Transformer model shows a strong downward trend in training loss and steadily improving token-level accuracy. The validation loss is substantially higher than the training loss across most epochs, which suggests either a distributional difference between the training and validation splits (or evaluation mismatch) or that further regularization (or more training) is required.

5.1.2 InceptionV3: The Visual Encoder

In the DrishtiXAI pipeline, **InceptionV3** serves as the system's "eyes." As the **Encoder**, its job is to translate a raw grid of pixels into a high-level mathematical language that the text-generating part of the model can understand.

From Pixels to Patterns Rather than looking at individual dots of color, the model works with feature maps created by InceptionV3. These maps identify complex visual concepts; like the distinct shape of a rickshaw, the texture of a sari, or the bustle of a local market. This compressed "summary" of the image is what allows the model to eventually write a descriptive Bengali sentence.

Why It Works for Bengali Scenes Capturing the essence of a Bangladeshi street or a rural landscape is difficult because these scenes are often "busy." InceptionV3 handles this complexity through:

- **Multi-Scale Processing:** It uses different filter sizes ($1 \times 1, 3 \times 3, 5 \times 5$) simultaneously. This allows it to see a tiny detail (like a vendor's product) and the broad context (the street itself) at the same time.
- **Preserving Detail:** The architecture is designed to avoid "information bottlenecks," ensuring that the small visual cues needed for a rich description are not "squashed" out before they reach the text decoder.
- **Efficiency:** The use of factorized convolutions (e.g., $1 \times n$ and $n \times 1$) allows the network to maintain the depth necessary for understanding complex scenes without becoming computationally prohibitive during the training of the 18,542 samples.

The Evolution of Inception: V1 to V3

The Inception family was created to solve a classic AI problem: how do you make a model more accurate without making it too slow or expensive to run? [25]

Inception V1: The Foundation V1 introduced the idea of parallel paths. By running different types of convolutions at once, the network could capture visual data at multiple sizes simultaneously. While powerful, its depth made it computationally expensive.

Inception V2: Smarter, Not Harder V2 refined the math. It replaced large 5×5 filters with two smaller 3×3 filters, which cover the same area but use fewer parameters. It also introduced **asymmetric convolutions** (like a 1×3 followed by a 3×1), which further slashed the processing cost while allowing the network to grow "wider."

Inception V3: The Current Standard V3 is the version we use because it is the most robust. It added Label Smoothing, a technique that stops the model from memorizing the training data too strictly, which is vital for helping it generalize to new, unseen Bengali images. Combined with optimized training algorithms, V3 offers state-of-the-art accuracy with a very manageable "footprint."

Performance and Practical Impact. With the progression from Inception V1 to Inception V3, the architecture achieved substantial improvements in classification accuracy on large-scale image datasets. Inception V3 demonstrated state-of-the-art performance with a significantly reduced error rate, highlighting the effectiveness of architectural factorization, regularization techniques, and optimized training strategies. Overall, the evolution of the Inception models reflects a shift toward architectures that are both computationally efficient and highly accurate for visual recognition tasks.

Justification: Attention Maps vs. Activation Maps

A key design choice in our architecture is using dynamic Attention Maps instead of static or post-hoc Activation Maps. While both provide visual explanations, they serve fundamentally different purposes in the context of sequence generation.

Dynamic Temporal Context. Activation Maps (e.g., Class Activation Mapping) are typically static and represent the regions of an image that were most influential for a single global classification label. In contrast, **Attention Maps** are dynamic and temporal. Because image captioning involves generating a sequence of tokens, the "salient" part of the image changes with every word. By using Bahdanau Attention, the model can shift its focus—for example, attending to a "person" while generating the subject and shifting to a "bicycle" when generating the object.

Interpretability of the Translation Process. Attention maps provide a direct "window" into the model's reasoning at each timestep $2t$.3 While an activation map shows which features were generally important to the InceptionV3 backbone, the attention map ($\alpha_{t,i}$) reveals the alignment between the Bengali vocabulary and the visual ROI. This allows us to verify if the model is truly "looking" at a rickshaw when it generates the word "রিকশা" (Rickshaw), providing a higher level of transparency and debugging capability during the evaluation of the 18,542 training samples.

Feature Integration vs. Feature Localization. Activation maps are often used for localization tasks to see where a network is looking. However, in our pipeline, the attention map does more than localize; it performs **feature integration**. It calculates the context vector c_t as a weighted sum of the ROI features. This ensures that the GRU

decoder receives a filtered, noise-reduced version of the image data tailored specifically to the word it is about to predict, rather than a generic activation signal.

Table 5.3: Comparison between activation maps and attention maps in image captioning.

Feature	**Activation Map**	**Attention Map (Ours)**
Nature	Static / Post-hoc	Dynamic / Generative
Granularity	Global image importance	Word-by-word alignment
Role	Visualizes CNN "interest"	Weights decoder's "focus"
Output	Heatmap of pixels	Context vector c_t for RNN

InceptionV3 Encoder + Bahdanau Attention + GRU Decoder

Justification for Exclusion of Transformer Decoder. A Transformer decoder was deliberately excluded from the InceptionV3 experiment to establish a distinct "Classic Baseline." The combination of InceptionV3 and an RNN (GRU) faithfully reproduces the architecture pioneered in *"Show, Attend and Tell"*, representing the state-of-the-art for the pre-Transformer era. By pairing InceptionV3 with a GRU rather than a Transformer, we isolate two distinct paradigms: the *Recurrent* approach (sequential processing with hidden states) versus the *Attention-only* approach (fully parallel processing with self-attention) used in our EfficientNet experiment. Mixing InceptionV3 with a Transformer would obscure this clear generational comparison and reduce the value of the experiment as a control baseline.

The InceptionV3 experiment implements the classic Show–Attend–and–Tell style encoder–decoder with additive (Bahdanau) attention. The training artifacts include pre-computed image features (cached .npy files), a tokenizer, and checkpoints.

The loss values show steady training loss improvement from 1.20 (epoch 0) down to 0.66 (epoch 7). Validation loss remains near 1.0 throughout the run and slightly increases toward later epochs (0.98 Ø 1.05), indicating a smaller but persistent generalization gap compared with training loss.

Table 5.4: Training metrics for the InceptionV3 model

Metric	First epoch	Final recorded epoch (7)
Training loss	1.20	0.66
Validation loss	1.05	1.05

Interpretation: The InceptionV3-based model attains much lower numerical loss values than the EfficientNet model. This numeric difference should not be interpreted as an absolute comparison of model quality because the two experiments use different decoder architectures and loss formulations, vocabulary sizes and decoding strategies (greedy vs stochastic sampling). Instead, focus on within-experiment trends: InceptionV3 shows consistent training improvement and a relatively stable validation loss, suggesting reasonable generalization for the trained regime.

5.1.3 Justification for Architecture Selection

The decision to use a Convolutional Neural Network (CNN) and specifically the **InceptionV3** architecture over other prominent models like ResNet, VGG, or Vision Transformers (ViTs) is rooted in structural and empirical considerations specific to the task of image captioning.

Inception vs. ResNet and VGG. While VGG models are simple and effective, their massive parameter count (138M+) and high memory usage make them computationally expensive for training on large-scale datasets. **ResNet** (Residual Networks) is highly efficient due to skip connections that solve the vanishing gradient problem; however, the Inception architecture offers a unique advantage for captioning via its **Inception modules**. By running parallel filters of different sizes ($1 \times 1, 3 \times 3, 5 \times 5$) in the same layer, InceptionV3 captures **multi-scale features**. This is critical for the BahdanauAttention mechanism: it allows the model to "look" at tiny details (using 3×3 filters) and broad environmental context (using 5×5 filters) simultaneously. This ensures that the attention module has access to both the specific objects (e.g., a "person") and their surrounding context (e.g., "sitting on a bench").

CNNs vs. Vision Transformers (ViT). While Vision Transformers (ViT) are currently state-of-the-art for many tasks, they present specific challenges for this project:

- **Data Hunger:** ViTs lack the "inductive bias" of CNNs; namely, the built-in understanding of spatial locality and translation invariance. Consequently, they require much larger datasets (millions of images) to learn what a CNN understands inherently.
- **Spatial Mapping:** For the "Show, Attend, and Tell" mechanism, the localized feature maps of InceptionV3 (structured as an 8×8 grid) provide a more intuitive spatial map. A ViT processes images as a sequence of global patches, which can sometimes lose the precise spatial "pointing" ability that an RNN decoder needs when focusing on a specific region of an image.

Empirical Rationale. Our preliminary trials confirmed that InceptionV3 struck the ideal balance between:

- **Feature Depth:** It provides a rich 2048-dimensional feature vector, which is more than enough to represent complex Bengali descriptions.
- **Spatial Granularity:** The 8×8 grid allows the attention model to effectively "point" to 64 distinct regions of the image, which is the perfect resolution for most real-world scenes.
- **Training Speed:** Optimized factorization (breaking down large filters into smaller ones) allowed us to train on 18,542 samples quickly without sacrificing the model's ability to "see" intricate details.

In summary, InceptionV3 remains the "Goldilocks" model for this project: it is deep enough to understand complex Bengali contexts but efficient enough to train without requiring server-grade supercomputers.

5.2 Qualitative Outputs and Attention

Both models have implementation differences, and these differences affect qualitative behavior:

- The EfficientNet model uses greedy decoding (argmax) during generation by default; captions are therefore deterministic given the same checkpoint.

- The InceptionV3 model uses stochastic sampling ('tf.random.categorical') at inference, producing diverse captions for the same image; for repeatable evaluation use the deterministic sampling or beam search.

Attention visualizations produced by the InceptionV3 pipeline are particularly useful to verify that the model focuses on sensible spatial regions while decoding.

5.3 Comparative Analysis and Caveats

Direct numeric comparison between the two runs is not advisable without normalizing the evaluation protocol; below we expand that caveat and summarize the observed differences in five concrete dimensions in Table 5.5.

5.3.1 Convergence

The EfficientNet-B4 + Transformer run shows a strong drop in training loss from 8.01 to 3.19 over the recorded epochs, with token-level training accuracy rising to 0.40; however its validation loss remains comparatively high (final recorded value 22.22). This pattern indicates fast in-run optimization on the training objective but a persistent training–validation gap.

The InceptionV3 + Bahdanau/GRU run attains much lower numerical losses (training loss 0.66 at the last logged epoch and validation loss 1.05). Within its own training regime the Inception experiment exhibits faster apparent convergence to a lower-loss plateau, but note the two runs use different loss scalings and vocabulary/tokenization which prevent direct numerical comparison of loss magnitudes.

5.3.2 Stability

Stability refers to how smoothly metrics evolve across epochs and how sensitive the run is to overfitting. The EfficientNet Transformer shows steady training improvement but a larger and more variable validation loss, suggesting either unstable generalization or an evaluation/preprocessing mismatch between training and validation. The InceptionV3 run shows steadier validation loss (a near-flat curve around 1.0) with smaller per-epoch validation fluctuations, indicating higher apparent validation stability under the current experimental settings.

5.3.3 Interpretability

Interpretability differs by architecture and by the visualization tools implemented: the InceptionV3 pipeline explicitly produces spatial attention visualizations coming from the Bahdanau additive attention. These visualizations map decoded tokens to image regions and are directly useful for human inspection of model focus. The EfficientNet Transformer uses token/self-attention in the decoder; unless cross-attention visualizations were implemented, its attention is less directly mapped to image pixels in the current outputs. In short: the InceptionV3 artifacts provide immediate spatial interpretability; the EfficientNet run may require additional tooling (cross-attention heatmaps or gradient-based saliency) to reach the same level of spatial explainability.

5.3.4 Decoding Behavior and Its Effects

Decoding choices materially affect evaluation and qualitative captions. The EfficientNet experiment was decoded deterministically (greedy/argmax) in the saved outputs, producing repeatable captions; the InceptionV3 experiment used stochastic sampling ('tf.random.categorical') during generation, yielding diverse captions for the same image. As a result, BLEU/CIDEr and other n-gram metrics computed from the two runs are not directly comparable unless the same deterministic decoding (greedy or beam search with fixed seeds) is used for both. Stochastic decoding also inflates apparent diversity (helpful for qualitative analysis) but reduces metric stability for single-run comparisons.

Table 5.5: Qualitative evaluation: concise comparison of convergence, stability, interpretability, decoding behavior, and error typology between the two experiments.

Dimension	**EfficientNet-B4 + Transformer**	**InceptionV3 + Bahdanau/GRU**
Convergence	Training loss drops strongly (8.01 → 3.19) with token accuracy reaching ∼0.40; validation loss remains high (∼22.22), indicating pronounced train–val gap.	Lower numerical training loss (1.20 → 0.66) with stable validation around ∼1.05; converges to lower-loss plateau within its regime.
Stability	Training metrics improve steadily but validation shows larger fluctuations and higher variance, suggesting possible overfitting or evaluation mismatch.	Validation curve is comparatively flat and less noisy (∼1.0), indicating higher apparent stability under current settings.
Interpretability	Decoder uses token/self-attention (transformer); cross-attention visualizations not provided by default; spatial explainability limited without extra tooling.	Produces spatial attention visualizations (Bahdanau) mapping tokens to image regions; immediately useful for human inspection.
Decoding Behavior	Deterministic (greedy/argmax) decoding for saved outputs. Repeatable but may yield terse captions; metrics stable for single-run evaluation.	Stochastic sampling (`tf.random.categorical`) for generation which yields diverse captions but increases metric variance and complicates comparisons.
Error Typology	Large train–val gap suggests lexical/tokenization mismatches, preprocessing issues, and overfitting; hallucinations and attribute errors expected.	Hallucinations and misattributed objects occur but easier to diagnose via attention maps; stochastic decoding can introduce repetition/truncation artifacts.
Recommended Follow-up	Verify tokenizer & preprocessing alignment; add cross-attention or saliency visualizations; apply targeted regularization and small-scale finetuning.	Recompute metrics under deterministic decoding (or aggregate samples); use attention maps for targeted error analysis and qualitative illustrations.

5.3.5 Limitations of Comparison

Experimental and Implementation Differences

The two experiments differ in multiple orthogonal dimensions (encoder backbone, decoder family, attention implementation, tokenizer/vocabulary, loss scaling and 'from_logits' settings, decoding strategy and evaluation scripts). Any direct numeric comparison (for example, of cross-entropy loss values or raw BLEU numbers obtained under different decoding policies) is therefore confounded by these implementation differences.

Evaluation Protocol and Metric Sensitivity

Evaluation metrics such as BLEU and CIDEr are sensitive to decoding determinism, vocabulary choices and tokenization. Stochastic decoding increases output variance and reduces the interpretability of single-run metric snapshots. Without a unified decoding and tokenization protocol, numeric metrics can be misleading.

Dataset Splits and Preprocessing

If training/validation/test preprocessing differs (image resizing, normalization, or text normalization/token filtering), observed validation gaps may reflect preprocessing mismatches rather than model capacity. We did not re-run training under a single unified pipeline.

5.4 Summary

Both the EfficientNet-B4 Transformer and the InceptionV3 + Bahdanau attention experiments show meaningful training progress and produce usable caption outputs. The numerical training traces differ in scale and trend because of architectural and implementation differences; the next step is to standardize the evaluation protocol (tokenizer, decoding, metric computation) and then compare BLEU/CIDEr on the same held-out test set. Qualitatively, attention visualizations from the InceptionV3 experiment are already useful for interpreting model focus; the EfficientNet transformer (with its larger embedding size) is a promising candidate for further tuning and checkpoint selection based on BLEU.

6 Discussion

This chapter synthesizes our experimental findings, performance metrics, and user feedback. We analyze the trade-offs between the two architectural paradigms; the **InceptionV3** baseline and the **EfficientNet** model and discuss the implications of training dynamics, qualitative errors, and user perceptions.

6.1 Challenges

Architectural Comparisons and Trade-offs Our experiments contrasted two distinct generations of image captioning.

- **The "Goldilocks" Baseline:** The InceptionV3 + Bahdanau Attention + GRU configuration proved to be the superior choice. It demonstrated higher stability and a significantly lower validation loss (≈ 1.05). Its greatest strength was interpretability; the spatial attention maps allowed us to visually verify that the model was "looking" at a rickshaw while typing the Bengali word for it.
- **The Transformer Struggle:** The EfficientNetB4 + Transformer model, while modern, struggled with our specific dataset size. Transformers are notoriously "data-hungry" and lack the built-in spatial understanding (inductive bias) of CNNs. With 18,542 samples, this model overfitted heavily, leaving the InceptionV3 pipeline as the more efficient and robust solution for this scale.

Training Dynamics and Generalization The training traces revealed a common hurdle in low-resource AI tasks: **Early Saturation**.

- The model hit its peak performance at Epoch 3 (validation loss: 0.9367).
- Even though the training loss kept dropping for another 20 epochs, the model was essentially just memorizing the training set rather than learning to "understand" images.
- By using strict early stopping and restoring weights from Epoch 3, we ensured the final system used for deployment was the one most capable of handling new, unseen photos.

Qualitative Error Analysis and Language Challenges The model's mistakes provided deep insight into the difficulty of bridging vision and the Bengali language.

- **Hallucination vs. Drift:** Sometimes the model "guessed" details (like saying "four boys" when there were five) based on what it had seen most often in training, rather than what was in the photo. The model would get the "gist" right (e.g.,

recognizing a field) but get the action wrong (e.g., saying someone is "running" instead of "crawling"), which penalized its accuracy scores even if the caption felt partially correct.

- **The Complexity of Bengali:** Unlike English, Bengali has a very rich system of inflections. A simple verb like "eating" can take many different forms depending on the context. Because our dataset was limited, the model occasionally struggled to pick the exact grammatically correct version of a word, leading to the "incomplete phrases" we observed in the outputs.

6.2 Threats to Validity

We discuss potential threats to the validity of our results and the steps taken to mitigate them. We organize the discussion using common validity categories from empirical software and ML research.

Internal validity Implementation bugs, uncontrolled randomness, and limited hyperparameter exploration can affect the internal correctness of our results. We mitigated these risks by using saved checkpoints, deterministic decoding when computing metrics, and by re-running key experiments after fixing obvious issues. Nonetheless, some experiments (e.g., Transformer runs) were expensive and therefore performed with a limited hyperparameter sweep; this may have reduced the chance of finding a stronger Transformer configuration.

External validity Our dataset (assembled from multiple sources and containing 18,539 image–caption pairs for training) covers many common scenes but is not exhaustive. Domain bias (photographic style, scene composition, cultural content) may limit generalization to other Bengali-speaking populations or application domains. The mobile app evaluation was performed on a limited set of devices and environments; performance and latency may differ on other hardware.

Construct validity Automatic metrics such as BLEU and ROUGE approximate lexical and n-gram overlap but do not fully capture semantic adequacy, cultural appropriateness, or fluency in Bengali. Tokenization and vocabulary design choices (e.g., how compound characters and inflections are handled) also affect measured performance. We therefore complemented automatic scores with qualitative examples and a small user study, but the human evaluation sample size (20 participants) is limited.

Conclusion validity Reported scores can be affected by variance across training runs, random seeds, and dataset splits. Where practical we restored the best validation checkpoint and report deterministic evaluation numbers; however, we did not perform a large number of independent restarts for every model due to compute constraints. As a result, some observed differences (especially between large models) may be sensitive to training variance.

Ethical and privacy considerations The dataset contains images drawn from public sources; we followed standard de-identification and licensing checks where applicable. User study participants provided consent for audio-recorded feedback. Despite this, cultural

biases present in training captions and safety risks from incorrect descriptions (particularly in assistive contexts) remain a concern.

Summary and mitigations To reduce these threats we combined multiple evaluation modalities (automatic metrics, qualitative inspection, attention visualizations, and a user study), used early stopping and checkpoint restoration to avoid overfitting, and documented dataset provenance and preprocessing decisions. Future work will expand dataset diversity, report confidence intervals across multiple runs, increase human-evaluation sample sizes, and explore stronger hyperparameter searches to reduce uncertainty.

6.3 Research Questions and Answers

This project answers five key research questions as mentioned in the Introduction section:

1. **Can attention-based models generate accurate Bengali captions with limited training data?**
   Our experiments demonstrate that with 18,539 training images, the InceptionV3-GRU model with Bahdanau attention achieved a mean BLEU score of 0.5157 and ROUGE-1 of 0.340, confirming that attention-based architectures can produce semantically meaningful Bengali captions even under resource constraints.

2. **Do attention visualizations improve user trust and help diagnose errors in Bengali captioning systems?**
   Attention heatmaps successfully highlighted image regions (e.g., focusing on a cyclist or baseball player) corresponding to generated words, making model decisions interpretable. Survey results show high clarity ratings (5.80/7), and attention maps helped identify failure modes such as hallucinations and semantic drift in complex scenes.

3. **How do different CNN encoders (EfficientNet-B4 vs. InceptionV3) compare in terms of caption quality, generalization, and efficiency for Bengali?**
   Comparative experiments revealed that EfficientNet-B4 with Transformer decoder suffered severe overfitting (validation loss reaching 22.2) due to limited data, while InceptionV3-GRU demonstrated superior stability (validation loss 1.05) and better XAI interpretability, making it more suitable for low-resource Bengali captioning.

4. **Which dataset strategies (augmentation, multiple captions per image, vocabulary design) work best for Bengali's complex grammar?**
   Our dataset combined multiple sources with varying caption densities (2-5.68 captions per image) totaling 65,912 captions. Early stopping at epoch 3 prevented overfitting, and multi-caption annotation improved linguistic diversity, though future work should focus on vocabulary expansion to reduce unknown tokens and semantic drift.

5. **Can TTS be integrated on mobile devices without sacrificing speed or reliability?**
   The mobile application successfully integrated on-device Bengali TTS functionality, enabling visually impaired users to hear captions. User surveys confirmed high supportiveness ratings (5.65/7) and overall satisfaction (5.95/7), validating that TTS integration enhances accessibility without compromising usability.

7 Conclusion and Future Work

7.1 Conclusion

This study investigated the creation of a Bengali picture captioning model that included CNN for feature extraction and RNN for natural language production. Despite working in a low-resource setting, the system was able to produce meaningful, human-readable captions and established the viability of multimodal learning for underrepresented languages like Bengali.

Bengali caption generation is a difficult task, as evidenced by the difficulties encountered, which range from dataset restrictions and computing constraints to linguistic complexity and evaluative biases. The combined effects of dataset imbalance, scarce training resources, and the lack of extensive Bengali benchmarks are reflected in moderate BLEU scores and variability in caption quality.

However, this research lays a solid platform for future improvement. The study emphasizes the need for stronger architectures, larger datasets, better annotation quality, and deeper attention mechanisms. This chapter identifies interesting future avenues for improving captioning systems' accuracy, detail, and linguistic fluency.

This paper advances Bengali AI research by highlighting problems, providing practical answers, and laying the groundwork for future multimodal systems. Future Bengali captioning systems can be highly expressive and contextually aware by increasing datasets, refining topologies, and incorporating multilingual pretrained models. This will improve accessibility, information retrieval, and richer digital communication.

7.2 Future Work

The limitations and insights discovered in this study suggest several future research areas. Developing these areas may result in more accurate, fluent, and deployable Bengali image captioning systems for many real-world applications. We will evaluate the performance of our proposed approach by translating a commonly used English image captioning dataset into Bengali.

7.2.1 Advancing Model Architectures

Recent vision tests have shown that transformer-based visual encoders, such as ViT and Swin Transformer [26] outperform classic CNN-LSTM architectures [27, 28]. Cascaded or modular neural networks, shown effective in low-resource contexts [29], offer a potential approach for structured feature extraction and caption modification. Furthermore, ensemble-based methods that combine several CNN or transformer backbones [30] might produce richer visual embeddings and lower object recognition uncertainty.

7.2.2 Improving Attention, Alignment, and Localization

Future research could concentrate on enhanced attention mechanisms, such as multi-head or hierarchical attention, to improve the model's capacity to emphasize salient regions during caption production. Incorporating region-based feature extraction approaches, such as using object recognition models like Faster R-CNN, could help to improve spatial awareness and fine-grained caption quality.

7.2.3 Dataset Expansion and Annotation Improvements

Increasing the dataset's visual and linguistic diversity is critical for enhancing generalization [14]. Improving annotation quality with more consistent syntax and descriptive information would increase the learning signal. Multiple captions per image should be used to improve linguistic diversity and reduce assessment variation, and category rebalancing may help to mitigate dataset-induced biases.

7.2.4 Leveraging Pretraining and Transfer Learning

When adjusted, pretrained visual encoders like EfficientNet, which have demonstrated efficacy in Bengali NLP and vision tasks [31]], may enhance feature representation even more. Better morphological and grammatical handling can be achieved by using multilingual or Bengali-specific language models, such as mBERT, XLM-R, or Bengali transformers [32]. Additionally, multimodal alignment for Bengali image–text pairs may be strengthened by contrastive pretraining techniques like CLIP.

7.2.5 Improving Training Efficiency

Memory-efficient training techniques, like mixed precision, gradient checkpointing, and tensor offloading, can facilitate the training of larger models on constrained hardware. The acceleration of large-scale experimentation can be achieved through distributed training that utilizes cloud-based GPUs or TPUs. Moreover, model compression methods such as pruning and quantization facilitate deployment on mobile or resource-limited platforms.

7.2.6 Mitigating Dataset and Model Bias

Future initiatives must integrate clear bias assessment measures to evaluate linguistic and visual discrepancies within the dataset. Class-balanced loss functions or balanced sampling techniques might enhance fairness across a variety of image types and lessen overfitting toward dominating object categories.

7.2.7 Enhancing Caption Fluency

Enhancing caption fluency may entail grammar-conscious decoding, the use of morphological rules, or hybrid neural-rule-based methodologies. Reinforcement learning methodologies utilizing semantic-oriented metrics as CIDEr, SPICE, or human input as rewards could enhance the semantic accuracy of generated captions.

7.2.8 Developing APIs for Organizational Integration

A crucial future objective is the creation of production-ready, scalable APIs that enable enterprises to include Bengali picture captioning into their operational processes. Such APIs

will provide automatic caption production without necessitating skill in machine learning. The framework must support domain-specific fine-tuning using an organization's own curated datasets, enabling adaptability to specialized sectors such as healthcare, agriculture, media archives, and e-commerce. A modular backend that accommodates customized encoders, tokenizer modifications, and vocabulary augmentation would significantly improve usability. Ultimately, security protocols, model versioning, and monitoring functionalities will be crucial for dependable large-scale implementation.

7.2.9 Quantitative Metrics for XAI Faithfulness

While our current evaluation relies on qualitative attention inspection and user surveys, a critical future direction is the implementation of formal, quantitative benchmarks for XAI. Future versions of DrishtiXAI could incorporate metrics such as **Faithfulness (Consistency)**, which measures how well the attention heatmaps correlate with the features most responsible for the model's prediction, and **Robustness**, which evaluates whether small perturbations to the visual input lead to logical changes in the explanation. Additionally, developing a "Ground Truth" attention dataset for Bengali visual contexts would allow for the calculation of **Pointing Game** scores, moving the evaluation from subjective interpretation to standardized, reproducible scientific benchmarks.

Bibliography

[1] Shenzhen Xunlong Software CO., Limited, "Orange pi zero 3 image."

[2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[3] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning (ICML)*, 2015.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] A. H. Kamal, M. A. Jishan, and N. Mansoor, "Textmage: The automated bangla caption generator based on deep learning," in *2020 International Conference on Decision Aid Sciences and Application (DASA)*, pp. 1–6, IEEE, 2020.

[6] M. Rahman *et al.*, "Chittron: An automatic bangla image captioning system," *Procedia Computer Science*, vol. 154, pp. 636–642, 2019.

[7] Central Intelligence Agency, "CIA World Factbook," 23 Jan 2024.

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[9] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[10] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[11] S.-H. Han, M.-S. Kwon, and H.-J. Choi, "Explainable ai (xai) approach to image captioning," *The Journal of Engineering*, vol. 2020, no. 13, pp. 589–594, 2020.

[12] R. K. Mohan, S. Sureshkumar, and V. Sivasubramaniam, "Explainable image captioning using cnn-cnn architecture and hierarchical attention," *arXiv preprint arXiv:2407.09556*, 2024.

[13] M. K. Kabir *et al.*, "Automatic bengali image captioning using efficientnet-transformer network," in *2023 22nd International Symposium on Communications and Information Technologies (ISCIT)*, pp. 110–115, IEEE, 2023.

[14] M. Faiyaz Khan, S. M. Sadiq-Ur-Rahman, and M. Saiful Islam, "Improved bengali image captioning via deep convolutional neural network based encoder-decoder model," *Algorithms for Intelligent Systems*, pp. 217–229, 2021.

[15] A. Bhuiyan, E. Hossain, M. M. Hoque, and M. A. A. Dewan, "Enhancing image caption generation through context-aware attention mechanism," *Heliyon*, vol. 10, no. 17, 2024.

[16] T. Deb *et al.*, "Oboyob: A sequential-semantic bengali image captioning engine," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 6, pp. 7427–7439, 2019.

[17] F. Hussain, R. Hussain, and E. Hossain, "Explainable artificial intelligence (xai): An engineering perspective," *arXiv preprint arXiv:2101.03613*, 2021.

[18] P. Natarajan and A. Nambiar, "Vale: A multimodal visual and language explanation framework for image classifiers using explainable ai and language models," *arXiv preprint arXiv:2408.12808*, 2024.

[19] B. Das, R. Pal, M. Majumder, S. Phadikar, and A. A. Sekh, "A visual attention-based model for bengali image captioning," *SN Computer Science*, vol. 4, no. 2, p. 208, 2023.

[20] H. Al Faraby, M. M. Azad, M. R. Fedous, M. K. Morol, *et al.*, "Image to bengali caption generation using deep cnn and bidirectional gated recurrent unit," in *2020 23rd international conference on computer and information technology (ICCIT)*, pp. 1–6, IEEE, 2020.

[21] M. A. Jishan, K. R. Mahmud, A. K. Al Azad, M. R. Ahmmad, B. P. Rashid, and M. S. Alam, "Bangla language textual image description by hybrid neural network model," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 2, pp. 757–767, 2021.

[22] Shenzhen Xunlong Software CO., Limited, "Orange pi zero 3 hardware details."

[23] Viso.ai, "Efficientnet: Complete guide to the sota computer vision model."

[24] GeeksforGeeks, "Efficientnet architecture."

[25] GeeksforGeeks, "Inception v2 and v3 - inception network versions."

[26] M. Hossain, G. Shahriar, M. Syeed, M. Uddin, M. Hasan, S. Shivam, and S. Advani, "Region of interest (ROI) selection using vision transformer for automatic analysis using whole slide images," *Scientific Reports*, vol. 13, p. 11314, 2023.

[27] G. Geetha, T. Kirthigadevi, G. G. Ponsam, T. Karthik, and M. Safa, "Image captioning using deep convolutional neural networks (cnns)," *Journal of Physics: Conference Series*, vol. 1712, p. 012015, 12 2020.

[28] S. Liu, L. Bai, Y. Hu, and H. Wang, "Image captioning based on deep neural networks," *MATEC Web of Conferences*, vol. 232, p. 01052, 2018.

[29] M. Islam, W. Rahman, M. Hossain, K. Roksana, I. Azpíroz, R. Diaz, I. Ashraf, and M. Samad, "Medicinal plant classification using particle swarm optimized cascaded network," *IEEE Access*, vol. 12, pp. 42465–42478, 2024.

[30] S. Sifat, M. Hossain, S. Tonny, B. Majumder, R. Mahajabin, and H. Shakhawat, "Android ransomware attacks detection with optimized ensemble learning," in *International Conference On Cybersecurity, Cybercrimes, And Smart Emerging Technologies*, pp. 41–53, 2022.

[31] M. Hasan, N. Alfaz, M. Alam, A. Rahman, H. Shakhawat, and S. Rahman, "Detection of Parkinson's disease from T2-weighted magnetic resonance imaging scans using EfficientNet-V2," in *2023 26th International Conference On Computer And Information Technology (ICCIT)*, pp. 1–6, 2023.

[32] F. M. Shah, M. Humaira, Jim, A. S. Ami, and S. Paul, "Bornon: Bengali Image Captioning with Transformer-based Deep learning approach," *arXiv*, 09 2021.