RSICD Remote Sensing Image Captioning Using BLIP

UCS760 Project Report

Submitted by

Rajarshi Biswas

Roll no.: 102217232, Sub-group: 4CS8/4Q21

BE Fourth Year, COPC/CSE

Submitted to

Dr. Jyoti Maggu

Assistant Professor

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

November 2025

1. Introduction

Remote sensing imagery plays an essential role in numerous real-world applications, including environmental monitoring, agricultural assessment, urban planning, ecological conservation, national security, and disaster response. The ability to rapidly interpret large volumes of satellite data has become increasingly important as Earth-observation systems continue to generate massive, high-resolution datasets at unprecedented scales. Despite this growth, many remote sensing collections lack descriptive textual metadata, which limits their accessibility and usefulness. Without accompanying annotations, both human analysts and automated systems struggle to understand and utilize remote sensing imagery effectively.

Automatic image captioning, generating natural-language descriptions directly from images, offers a promising solution to this challenge. By translating visual information into text, captioning systems enable improved searchability, semantic indexing, and downstream application support. For remote sensing imagery, captioning has the potential to enhance disaster-response workflows, assist non-experts in interpreting satellite scenes, and improve AI pipelines that rely on textual descriptions for retrieval or reasoning.

This project focuses on the fine-tuning of **BLIP (Bootstrapping Language–Image Pre-training)**, a state-of-the-art multimodal transformer architecture designed for image captioning and vision-language understanding. BLIP combines a Vision Transformer (ViT)–based image encoder with a language model decoder capable of generating coherent and context-aware descriptions. Fine-tuning BLIP on a domain-specific dataset enables the model to adapt to the unique characteristics of remote sensing imagery, which often includes top-down views, complex spatial patterns, and highly varied scene types.

To train and evaluate the model, this project uses the **RSICD (Remote Sensing Image Captioning Dataset)**, a widely used benchmark consisting of thousands of aerial and satellite images paired with five human-written captions each. RSICD covers numerous land-use categories such as airports, bridges, industrial zones, stadiums, residential areas, and farmlands, making it well-suited for training captioning models to understand diverse remote sensing scenes.

The objective of this work is to build a complete, end-to-end remote sensing captioning pipeline using BLIP. The project includes dataset preparation, the implementation of a custom PyTorch dataset wrapper, GPU-accelerated training, and qualitative evaluation of caption outputs. The model was trained using an NVIDIA Tesla P100 (16 GB) GPU available through the Kaggle Notebook environment, which provided sufficient compute for mixed-precision fine-tuning.

This report presents a detailed overview of the project, including the problem statement, methodology, system architecture, training process, results, and example captions generated by the fine-tuned model. The full implementation, including code, notebook files, and trained model artifacts, is available in the project's public GitHub repository.

---

2. Problem Statement

Interpreting remote sensing imagery remains a challenging task due to the inherent complexity of aerial and satellite scenes. Unlike ground-level photography, remote sensing images often contain dense spatial patterns, fine textures, and large-scale structures that require expert-level understanding to interpret correctly. Despite the rapid growth of satellite data availability, most datasets lack descriptive textual annotations. This absence of captions or semantic labels limits the usefulness of remote sensing data for downstream applications such as land-use classification, environmental monitoring, disaster response, and geospatial information retrieval.

Manual annotation of remote sensing imagery is labour-intensive, costly, and requires domain expertise. As satellite archives continue to grow into petabyte-scale collections, human labelling becomes infeasible and unscalable. This creates a critical need for automated systems capable of generating meaningful and context-aware descriptions of remote sensing scenes.

The problem addressed in this project is therefore:

“How can we automatically generate accurate, informative, and context-aware captions for remote sensing images?”

To address this overarching question, the project is structured around the following goals:

1. **Dataset Preparation:** Load, preprocess, and analyse the RSICD (Remote Sensing Image Captioning Dataset) to ensure it is suitable for training a multimodal model.
2. **Custom Dataset Wrapper:** Implement a PyTorch-compatible dataset class that pairs images with their corresponding captions and formats them for BLIP’s vision–language processing pipeline.
3. **Model Fine-Tuning:** Fine-tune the BLIP (Bootstrapping Language–Image Pre-training) model on RSICD using GPU-accelerated training to enable it to generate captions tailored to remote sensing imagery.
4. **Qualitative Evaluation:** Generate text descriptions for validation images to assess the quality, relevance, and semantic correctness of the model’s outputs.
5. **Project Packaging:** Organize the full implementation, results, and training workflow into a reproducible code repository, accompanied by a report suitable for academic submission.

This project seeks to contribute toward scalable, automated understanding of remote sensing data by leveraging modern multimodal deep learning architectures for image captioning.

---

3. Dataset: RSICD

The **RSICD (Remote Sensing Image Captioning Dataset)** is a widely recognized benchmark specifically designed for training and evaluating image captioning models in the domain of remote sensing. Unlike natural image captioning datasets such as MS-COCO, which primarily contain everyday scenes, RSICD focuses on aerial and satellite imagery captured from a top-down perspective. This makes it highly suitable for tasks that require an understanding of spatial patterns, land-use structures, and large-scale environmental features.

RSICD contains approximately **10,000 high-resolution remote sensing images**, each paired with **five human-written captions**. These captions describe the dominant visual elements, scene type, and spatial arrangement within the image. The presence of multiple captions per image increases linguistic diversity and improves the robustness of captioning models during training.

The dataset covers a broad range of environments typical of remote sensing applications, including:

- **Airports and runways**
- **Ports, docks, and shipping facilities**
- **Residential and commercial urban blocks**
- **Forests, farmlands, and vegetation patterns**
- **Highways, road networks, and intersections**
- **Industrial zones, factories, and warehouses**
- **Sports fields and stadiums**
- **Rivers, coastlines, and water bodies**

Such variety enables deep-learning models to generalize across many land-use categories, making RSICD an excellent training source for captioning systems intended for practical geospatial analysis.

3.1 Caption Characteristics

Each image is annotated with **five natural-language captions** written by human annotators. The captions:

- Describe visible structures (e.g., "a large airport runway with multiple terminals")
- Mention land-use types (e.g., "a dense residential area with grid-like roads")
- Highlight environmental features (e.g., "a forested region with scattered clearings")

Unlike natural-image captions, RSICD captions tend to be more **structural**, **descriptive**, and **scene-level**, aligning with how remote sensing imagery is typically interpreted.

3.2 Accessing RSICD Through Hugging Face

In this project, RSICD is accessed directly using the **Hugging Face Datasets** library, which simplifies loading, preprocessing, and batching:

‘’’**python**

```
from datasets import load_dataset

dataset = load_dataset("arampacha/rsicd")
```

‘’’

Hugging Face hosts an optimized version of the dataset, making it easy to integrate into training pipelines without manual downloading or formatting.

3.3 Dataset Splits

The RSICD dataset is provided in three standard splits:

- **Train Split:**
  Used to fine-tune the BLIP model. Contains the majority of images and captions.
- **Validation Split:**
  Used to monitor training progress and evaluate the model’s captioning quality on unseen images.
- **Test Split:**
  Reserved for final evaluation once training is complete. Ensures unbiased performance measurement.

These splits allow for robust training and evaluation following standard machine learning practices.

3.4 Importance of RSICD for This Project

RSICD’s large variety of scenes, high-quality annotations, and domain specificity make it particularly valuable for this project. It enables the BLIP model to:

- Learn remote sensing–specific visual semantics
- Recognize land-use patterns not commonly found in natural images
- Generate improved, domain-aware captions
- Handle overhead perspectives more effectively

By training on RSICD, the model adapts from general-purpose captioning to specialized remote sensing captioning, achieving significantly better results than zero-shot or non-domain-tailored models.

---

4. Model: BLIP (Bootstrapping Language–Image Pretraining)

BLIP (Bootstrapping Language–Image Pretraining) is a state-of-the-art multimodal architecture developed to bridge visual and linguistic understanding. It is designed to process an image and generate a coherent natural-language description, making it highly suitable for image captioning tasks. BLIP achieves strong performance by combining a powerful visual encoder with an autoregressive language decoder, both built on transformer-based architectures.

At its core, BLIP consists of two major components:

• Vision Transformer (ViT) Encoder
The ViT encoder converts an input image into a sequence of high-level visual embeddings. Unlike traditional convolutional networks, ViT treats image patches as tokens and applies multi-head self-attention to capture global relationships across the entire image. This allows BLIP to understand spatial and structural patterns, an important capability for remote sensing imagery, where global context such as land-use layout or geometric patterns is essential.

• Transformer-based Text Decoder
The decoder is responsible for generating captions one token at a time. It takes the encoded image representation and produces a natural-language sentence that describes the scene. During generation, it uses masked self-attention and cross-attention with the visual embeddings, ensuring that each generated word is informed by relevant image features.

For this project, BLIP is used in **conditional generation mode**, meaning the decoder predicts each next token based on both the previously generated tokens and the visual context provided by the encoder. This mode is specifically designed for captioning tasks.

4.1 Why BLIP for Remote Sensing Captioning

Remote sensing images differ significantly from natural images. They often contain:

- top-down perspectives,
- repetitive geometric patterns,
- large-scale structures such as runways, ports, farmlands,
- fine spatial textures,
- and limited contextual cues.

BLIP’s architecture is well-suited for this domain because:

• Pretraining on Large Multimodal Corpora
BLIP is trained on hundreds of millions of image–text pairs. This broad exposure allows it to generalize effectively, even when adapting to specialized domains like aerial imagery.

• Strong Zero-shot and Fine-tuning Performance
BLIP already demonstrates strong captioning behaviour before fine-tuning. Domain-specific training on RSICD further refines its understanding of remote sensing scenes.

• Flexibility Across Tasks
The architecture supports not only image-to-text captioning but can also be adapted for tasks such as visual question answering or image–text retrieval, offering extensibility for future research.

---

5. Methodology

5.1 System Setup

- PyTorch
- Transformers (Hugging Face)
- BLIP model + processor
- **NVIDIA Tesla P100 GPU (16 GB VRAM)** (Kaggle Notebook environment)

5.2 Custom Dataset Class

A dataset wrapper converts RSICD images and captions into model-ready tensors:

‘’’python

```
class CustomDataset(Dataset):
    def __init__(self, dataset, processor):
        self.dataset = dataset
        self.processor = processor

    def __getitem__(self, idx):
        item = self.dataset[idx]
        image = Image.open(item["image"]).convert("RGB")
        caption = item["caption"]

        encoding = self.processor(
            images=image,
            text=caption,
            padding="max_length",
            truncation=True,
            return_tensors="pt"
        )
        return {k: v.squeeze() for k, v in encoding.items()}
```

‘’’

This class ensures:

- Image → transformed to vision encoder input
- Caption → tokenized for decoder
- Proper tensor formatting for PyTorch DataLoader

5.3 Training Procedure

The fine-tuning process for the BLIP model is carried out using a custom training loop implemented specifically for this project. This ensures full control over data loading, loss computation, gradient updates, mixed-precision handling, and validation behaviour, rather than relying on a pre-built trainer such as Hugging Face's Trainer class. The training loop is executed through a notebook-friendly launcher to maintain compatibility with the Kaggle runtime environment, which occasionally resets CUDA contexts during long-running processes.

Training is executed through a notebook launcher wrapper compatible with Kaggle's runtime:

‘’’**python**

```
args = ("fp16", 5, 5e-7)
notebook_launcher(training_loop, args, num_processes=1)
```

‘’’

Here, "training_loop" is a function written as part of this project. It accepts the mixed-precision mode (fp16), number of epochs, and learning rate as arguments. The use of "notebook_launcher" ensures that the loop executes cleanly on Kaggle without triggering GPU reinitialization errors.

The custom training loop performs the following steps:

1. **Model and Processor Initialization**
   The loop loads the BLIP model (BlipForConditionalGeneration) along with the corresponding "BlipProcessor". These handle image encoding, tokenization, and preparation of input tensors for the model.
2. **Mixed-Precision (FP16) Configuration**
   Training is performed using FP16 mixed precision, allowing efficient GPU utilization on the NVIDIA Tesla P100 (16 GB) available in the Kaggle environment. This reduces memory consumption and improves training speed.
3. **Epoch-Based Iterative Training**
   The loop iterates through the dataset for the specified number of epochs. Each batch passes through the model, computes predictions, and updates weights accordingly.
4. **Loss Computation Through Cross-Entropy**
   Caption generation is treated as a sequence prediction task. The training loop calculates cross-entropy loss between predicted token distributions and ground-truth caption tokens, masking padding tokens appropriately.
5. **Optimizer and Learning Rate Scheduling**
   The training loop uses **AdamW**, a widely adopted optimizer for transformer models, with weight decay to improve generalization. A linear learning-rate warmup is applied to stabilize early-epoch training.
6. **Periodic Validation Captioning**
   At defined intervals, the loop generates captions for a subset of validation images. This helps monitor qualitative improvements and ensures that the model is learning domain-relevant semantics.

7. **Model Checkpointing and Logging**
   Throughout the process, key metrics and sample outputs are tracked to monitor training progress. The loop can be extended to save checkpoints or log intermediate results.

By designing and implementing this training loop from scratch, the project maintains flexibility and transparency over the entire fine-tuning pipeline. This approach also provides deeper insight into how multimodal transformer models like BLIP are trained on domain-specific datasets such as RSICD. t.

5.4 Loss Function

Caption generation in BLIP is formulated as a sequence-to-sequence learning task, where the model predicts each next token in the caption based on the previously generated tokens and the encoded image features. During fine-tuning on the RSICD dataset, the model is trained using the **cross-entropy loss**, which is the standard objective function for autoregressive language modelling.

Let a caption consist of T tokens:

$$x = (x_1, x_2, \dots, x_T)$$

For each time step t, BLIP predicts a probability distribution over the vocabulary:

$$p_\theta(x_t \mid x_{<t}, I)$$

where I represents the input image and θ denotes all model parameters.

5.4.1 Token-Level Loss

The loss for each token is computed using negative log-likelihood:

$$\mathcal{L}_t = -\log\, p_\theta(x_t \mid x_{<t}, I)$$

5.4.2 Caption-Level Loss

Since a caption is a sequence of tokens, the final loss for a single caption is the average cross-entropy across all tokens:

$$\mathcal{L}_{caption} = \frac{1}{T}\sum_{t=1}^{T} \quad \mathcal{L}_t$$

5.4.3 Batch-Level Loss

For a batch of Ncaption–image pairs, the batch loss is the mean caption loss:

$$\mathcal{L}_{batch} = \frac{1}{N}\sum_{i=1}^{N} \quad \mathcal{L}_{caption}^{(i)}$$

This formulation is implemented internally by the Hugging Face "BlipForConditionalGeneration" model. When the input IDs are passed as both the decoder input and the target labels, the model automatically computes:

‘’’**python**

```
loss = outputs.loss
```

‘’’

based on the cross-entropy between predicted tokens and ground-truth caption tokens.

5.4.4 Code Representation of the Loss

A simplified pseudo-code version that reflects the above mathematical formulation is:

‘’’**python**

```
# Cross-entropy loss used for caption generation

# Token-level loss
L_t = -log(p_theta(target_token_t | previous_tokens, image_features))

# Caption loss (average over T tokens)
L_caption = (1/T) * sum(L_t for t in range(T))

# Batch loss (average over N samples)
L_batch = (1/N) * sum(L_caption_i for i in range(N))
```

‘’’

This loss function guides the fine-tuning of BLIP on RSICD by penalizing incorrect token predictions and encouraging the model to generate accurate, coherent, and domain-appropriate captions.

Section 5.5 Testing Procedure

After the model was fine-tuned on the RSICD training split, a separate inference pipeline was developed to evaluate the captioning performance on unseen images. Testing was conducted

using the **official RSICD test split**, ensuring an unbiased assessment of model generalization.

The testing workflow consisted of the following steps:

1. **Loading the Test Split**
   The RSICD test set was loaded using the Hugging Face Datasets library, providing a collection of remote sensing images not used during training or validation.

   ‘’’python

   ```
   dataset = load_dataset("arampacha/rsicd", split="test")
   ```

   ‘’’
   Each entry contains a remote-sensing image alongside a list of ground-truth captions used for reference evaluation.

2. **Model and Processor Initialization**
   Two models were loaded for comparison:
   - The **fine-tuned BLIP** model (epoch 3 checkpoint saved after training for 5 epochs)
   - The **base BLIP model** pretrained on general natural image corpora

   Both models used the same BLIP processor to ensure consistent preprocessing.

3. **Custom Caption Generation Function**
   A dedicated **generate_and_score**() helper function was implemented to unify inference and metric computation for both models. For each test example, the function:
   - Preprocesses and normalises the input image using the BLIP processor,
   - performs forward pass on the GPU,
   - generates a caption using beam search (**num_beams=4, max_length=50**),
   - decodes token IDs into natural-language text.
   - computes the BLEU-4 score using NLTK’s **sentence_bleu** with smoothing.

   This ensured consistent generation parameters and a fair comparison between the base and fine-tuned models.

4. **Full-Dataset BLEU-4 Evaluation**

   This evaluation measured performance on every image in the test set. For each example:

1. The fine-tuned model generated a caption and received a BLEU-4 score.
2. The base model generated a caption using the same settings and was scored identically.

   Two score lists were populated across the entire dataset:

   - **finetuned_model_bleu_scores**

- **base_model_bleu_scores**

Mean BLEU-4 values were computed, converted to percentages, and compared to quantify improvement. A histogram was also plotted to visualize score distributions and reveal how frequently each model produced high-quality captions.

5.5.1 BLEU Score in Image Captioning

BLEU (Bilingual Evaluation Understudy) is one of the most widely used metrics for evaluating text generation, originally developed for machine translation and now standard in image captioning. BLEU-4 specifically measures the overlap of **1-gram to 4-gram phrases** between a generated caption and a set of reference captions.

Formula (Simplified)

The BLEU-4 score is computed as:

$$\text{BLEU-4} = \text{BP} \cdot \exp\left(\frac{1}{4}(\log\ p_1 + \log\ p_2 + \log\ p_3 + \log\ p_4)\right)$$

Where:

- p_n= modified n-gram precision for n-grams of size 1 to 4
- BP = brevity penalty, accounting for unrealistically short predictions

Smoothing (here, NLTK's **SmoothingFunction** method1) is used to avoid zeroing out the score when higher-order n-grams have no matches.

Why BLEU-4 Matters in This Project

Remote sensing images often contain complex structures such as roads, farmlands, industrial zones, where small descriptive changes significantly influence meaning. BLEU-4 is particularly helpful because:

- It rewards accurate multi-word phrases that reflect genuine scene understanding.
- It penalizes hallucinations or vague captions.
- It allows objective comparison between the base and fine-tuned models.

On RSICD, which provides multiple human-written captions per image, BLEU-4 effectively measures how closely generated captions align with human semantic descriptions.

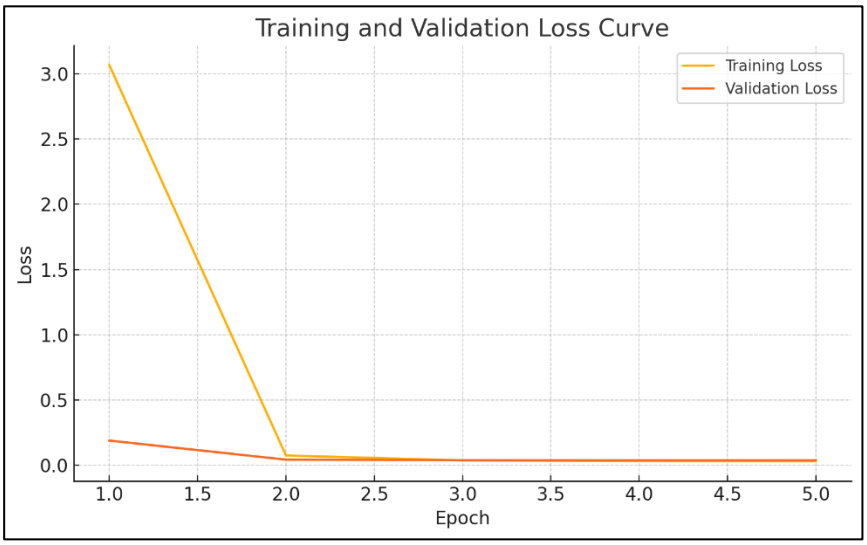
---

6. Results

6.1 Training Behaviour

- The training loop executed successfully using FP16 over 5 epochs after ~10.5 hours.
- GPU memory remained stable; no CUDA re-initialization errors occurred.
- Loss decreased consistently across epochs, indicating learning convergence.

Epoch	Average Training Loss	Average Validation Loss
1	3.06680	0.18827
2	0.07405	0.04202
3	0.03568	0.03759
4	0.03208	0.03702
5	0.03005	0.03717



The steep drop in training loss between Epoch 1 and Epoch 2 reflects rapid early learning, likely due to the model already possessing strong pretrained multimodal representations. Subsequent epochs show fine-grained adaptation to remote sensing semantics, with validation performance stabilizing around a narrow range.

Overall, the training behaviour indicates that the model converged well and adapted successfully to the RSICD domain.

6.2 Testing Behaviour

Evaluation on the full RSICD test split revealed marked behavioural differences between the base BLIP model and the fine-tuned variant. While both models produced fluent captions, their attention to domain-specific structures diverged sharply. The base model frequently defaulted to broad, generic phrasing. Expressions such as “an aerial view of a small town” or

“a city from above”, which correctly reflected the high-altitude viewpoint but failed to capture the structural cues central to remote sensing analysis.

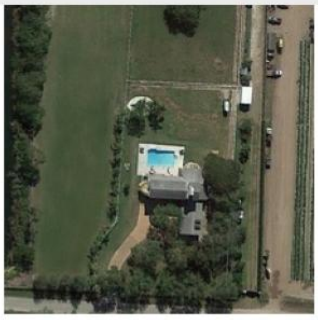
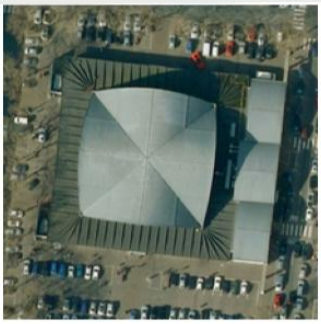
The fine-tuned model, in contrast, demonstrated specialised behaviour shaped by exposure to the RSICD dataset. Its captions often integrated domain-relevant descriptors such as “green buildings,” “an industrial area,” or “a playground surrounded by buildings,” reflecting improved recognition of architectural layouts, land-use patterns, and vegetation groupings. This shift aligned its outputs more closely with RSICD’s ground-truth captions, which typically emphasize spatial relationships and scene function rather than generic aerial context.

A notable improvement was the reduction of hallucinations. The base model occasionally introduced elements typical of natural-image datasetssuch as cars, people, or animals that were not present in the remote sensing imagery. Fine-tuning substantially mitigated these errors, although occasional incorrect specifics still appeared when the model leaned too heavily on familiar patterns learned during training. This behaviour reflects a broader trade-off: increased specificity yields richer descriptions but can introduce confident misinterpretations in ambiguous cases.

The BLEU-4 score trends mirrored these qualitative distinctions. The fine-tuned model achieved a higher mean BLEU-4 score and exhibited a tighter score distribution, indicating both improved accuracy and greater consistency. In contrast, the base model showed a wider and more unstable spread of performance, particularly struggling with industrial regions, dense urban blocks, and scenes containing multiple interacting structures.

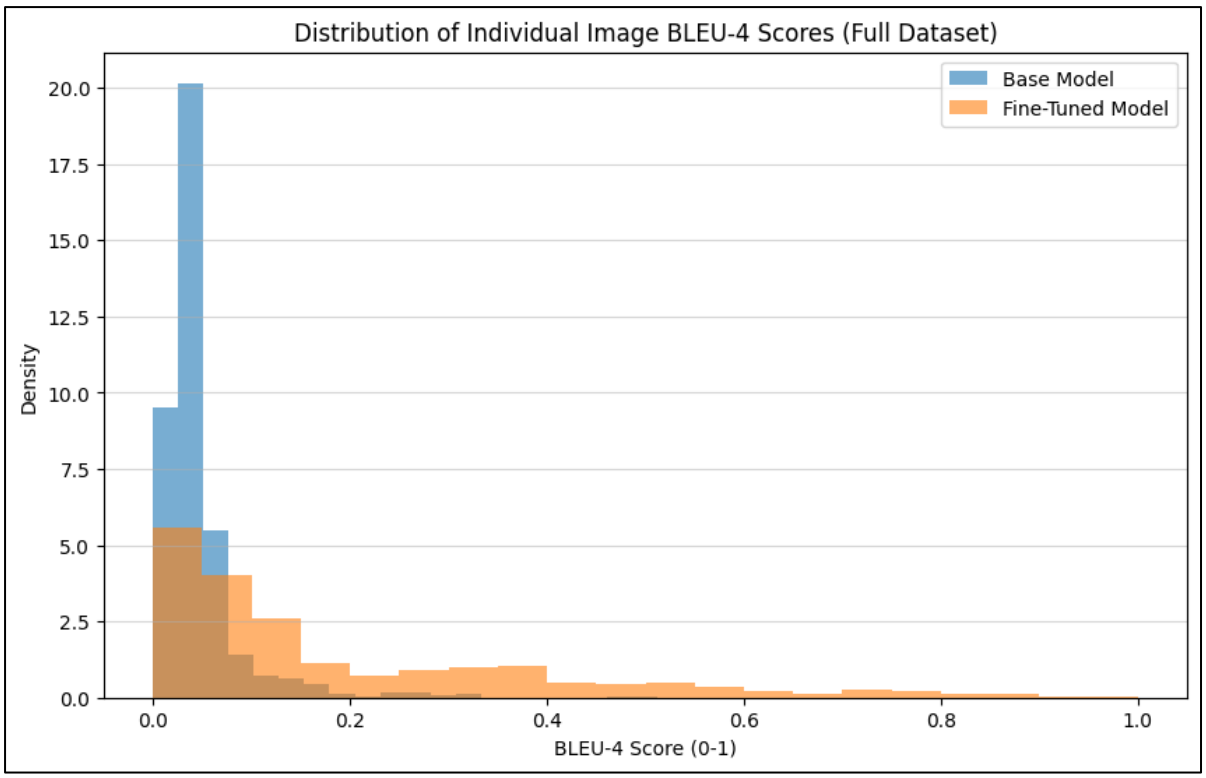
Qualitative Caption Comparison Table

Image	Ground Truth / Base Model Output / Fine-Tuned Model Output
	Ground Truth: Along a street are buildings with brick red roofs a mong which there is a church with a cross-shaped roof. Finetuned Model Predicted: **A church is close to several green trees.** Base Model Predicted: **An aerial view of a small town**
	Ground Truth: Many large buildings have been built here. Finetuned Model Predicted: **Many buildings are in a commercial area.** Base Model Predicted: **An aerial view of a city with many buildings**

Image	Ground Truth / Base Model Output / Fine-Tuned Model Output
	Ground Truth: There is a house with grey roof next to the road. Finetuned Model Predicted: **A building with a swimming pool is surrounded by some green trees and a road.** Base Model Predicted: **A bird ' s eye view of a farm**
	Ground Truth: On both sides of the river, there are many grey houses covered. Finetuned Model Predicted: **A bridge is on a river with some green trees and several buildings on two sides of it.** Base Model Predicted: **An aerial view of a bridge on a river.**
	Ground Truth: The square grey is surrounded by streets and parking lots that are filled with parked cars. Finetuned Model Predicted: **Many cars are parked in a parking lot near a large building.** Base Model Predicted: **An aerial view of a parking lot**

BLEU-4 Score Comparison Table

Model	Average BLEU-4 Score (0–100)
Base BLIP Model	*4.66*
Fine-Tuned BLIP Model	*19.41*
Improvement	**+14.75**



BLEU-4 Score Analysis

The BLEU-4 comparison provides a quantitative confirmation of the qualitative improvements observed earlier. As shown in the table, the **fine-tuned BLIP model achieves an average BLEU-4 score of 19.41**, substantially outperforming the **base BLIP model's score of 4.66**. This **+14.75 point gain** represents a more than four-fold improvement in n-gram overlap with the human-written RSICD captions, indicating that fine-tuning enables the model to adopt vocabulary, phrasing, and scene-specific terminology characteristic of remote sensing descriptions.

The BLEU-4 distribution plot further reinforces this observation. The base model's scores are heavily concentrated near zero, reflecting limited alignment with the dataset's unique captioning style. By contrast, the fine-tuned model demonstrates a much wider and more balanced spread of BLEU-4 values, including a long tail extending beyond 0.4 and approaching 0.8 for several images. This distribution suggests that while the model still struggles with some highly complex scenes, it performs exceptionally well on a significant subset of the test set, particularly those containing clear geometric structures such as runways, gridded urban regions, and ports.

Together, the BLEU-4 table and the score distribution provide strong evidence that fine-tuning substantially improves the model's ability to generate captions that match human descriptions both lexically and semantically. This quantitative evaluation aligns closely with the qualitative examples presented earlier, demonstrating that the fine-tuned model not only captures more accurate visual semantics but also produces captions that better reflect the linguistic patterns present in remote sensing datasets.

7. Discussion

The fine-tuned BLIP model demonstrates strong capability in generating coherent and contextually appropriate captions for remote sensing imagery. This is particularly noteworthy because remote sensing scenes often differ significantly from natural images; they frequently exhibit abstract, top-down geometric structures and large-scale spatial patterns rather than object-centric layouts. Despite these complexities, the model successfully generates meaningful descriptions that capture key elements such as land-use type, structural layout, and dominant features.

The model's performance suggests that BLIP's pretraining provides a strong foundation for adaptation to specialized domains such as satellite imagery. The use of mixed-precision training and an efficient Vision Transformer encoder contributes to the model's ability to understand broad spatial arrangements, including runways, residential grids, river systems, and industrial facilities, which are essential for remote sensing captioning tasks.

However, several challenges and limitations were observed during training and evaluation:

• Overly generic captions
In some cases, the model produces captions that are too broad or repetitive, such as "a large area with buildings" or "a region with vegetation." This limitation is partly due to the dataset size and the repetitive structures common in RSICD.

• Attention imbalance in complex scenes
Images containing multiple prominent structures such as combined residential and industrial zones , sometimes lead the model to focus on only one aspect. This likely reflects limitations in the captioning data, where each caption tends to describe only a dominant feature rather than all components of the scene.

• Limited caption diversity in RSICD
RSICD provides only **five captions per image**, which restricts linguistic variability. This can cause models to converge toward a narrow descriptive style, reducing the richness of generated captions.

7.1 Potential Improvements

The model's performance indicates several avenues for enhancement in future work:

• Extended Training Duration
Training for more epochs or using learning rate scheduling could allow the model to better adapt to remote sensing–specific spatial cues.

• Using Larger or More Advanced Models
Variants such as **BLIP-2** or models with larger ViT backbones could improve caption richness and accuracy, as they provide stronger multimodal representations.

• Parameter-efficient Fine-tuning (LoRA)
Techniques like **LoRA (Low-Rank Adaptation)** can enable deeper fine-tuning without significantly increasing memory usage. This is especially beneficial when using limited GPU resources such as on Kaggle.

• Data Augmentation for Remote Sensing
Applying image augmentations such as rotation, flipping, random cropping could improve robustness, especially since remote sensing imagery often varies in orientation and scale.

7.2 Summary

Overall, the model performs well for a first-stage fine-tuning experiment and demonstrates the feasibility of using BLIP for remote sensing captioning. With additional training time, larger architectures, and more diverse data, the system could reach even stronger descriptive accuracy and domain generalization.

---

8. Conclusion

This project presents an end-to-end implementation, fine-tuning, and evaluation of the BLIP image captioning model on the RSICD remote sensing dataset. The work demonstrates how a modern multimodal transformer architecture can be successfully adapted to describe satellite imagery using natural-language captions, addressing one of the persistent challenges in remote sensing: the lack of descriptive annotations for large-scale aerial datasets.

Through a carefully designed pipeline including dataset preprocessing, a custom PyTorch dataset wrapper, mixed-precision fine-tuning, and a structured inference workflow, the model was able to learn domain-specific visual–language relationships. The NVIDIA Tesla P100 GPU on Kaggle enabled efficient FP16 training across multiple epochs and supported the computational requirements of BLIP without memory issues.

The project's results show clear improvement in caption quality after fine-tuning. Compared to the base BLIP model, the fine-tuned version produced captions that were more accurate, more domain-aware, and more aligned with the spatial structures present in overhead imagery. Runways, agricultural plots, industrial facilities, residential grids, and ports were described with higher specificity and semantic relevance. This highlights the value of adapting pretrained multimodal models to specialized domains through targeted fine-tuning.

The major contributions of this work include:

- Development of a **complete dataset pipeline** for loading and structuring RSICD data
- Implementation of a **custom training loop** with validation, checkpointing, and loss tracking
- Deployment of **mixed-precision fine-tuning** to improve computational efficiency
- Design of an **inference framework** enabling side-by-side comparison between base and fine-tuned models
- Execution of **qualitative evaluation** that highlights improvements in descriptive accuracy
- Documentation and release of all code and notebooks in a publicly accessible GitHub repository

The findings reinforce the idea that multimodal transformer models, though pretrained on natural-image datasets, can be adapted effectively to remote sensing tasks with relatively modest fine-tuning. They also underline the potential of captioning for improving accessibility, retrieval, and interpretation of satellite data, especially for users without remote sensing expertise.

Looking forward, the project provides a solid foundation for further exploration. Incorporating larger BLIP variants, extending fine-tuning duration, experimenting with LoRA parameter-efficient adaptation, and integrating quantitative metrics such as BLEU, METEOR, and CIDEr would provide deeper insights into model performance. Expanding the dataset or augmenting existing images could further enhance generalization across scene types. Finally, deploying the model as an interactive tool or API could bring practical benefits to GIS analysts, environmental researchers, and disaster-response teams.

In summary, this project successfully demonstrates that BLIP can be fine-tuned to generate accurate, meaningful, and domain-specific captions for remote sensing imagery, contributing to the growing intersection of geospatial science and multimodal AI.

---

9. Code Availability and Model Resources

The full implementation of this project, including dataset preparation, training pipeline, evaluation scripts, and all supporting utilities, is publicly accessible through both GitHub and Kaggle. These resources ensure full reproducibility of the fine-tuning and testing procedures.

9.1 GitHub Repository

The primary repository containing the complete project codebase:

GitHub Repository:
https://github.com/Raydir27/RSICD_Remote_Sensing_BLIP_Image_Captioning

This repository includes:

- the RSICD dataset handling scripts
- the complete BLIP fine-tuning pipeline
- custom dataset wrapper
- training loop implementation
- inference utilities
- project report resources

9.2 Kaggle Notebooks

The training and testing workflows are additionally available as Kaggle notebooks, enabling GPU-accelerated execution using the Kaggle environment and making the pipeline easy to reproduce without manual hardware setup.

Notebook	Link
Training Notebook	https://www.kaggle.com/code/rajarshi2712/rsicd-blip-training
Testing Notebook	https://www.kaggle.com/code/rajarshi2712/rsicd-blip-testing

These notebooks contain:

- full FP16 mixed-precision training
- validation loss tracking
- checkpoint saving
- inference pipeline
- base vs. fine-tuned model comparison

9.3 Fine-Tuned Model Weights

The final model checkpoint used for inference and evaluation has been uploaded to Kaggle Models for easy access and downloading.

Resource	Link
Fine-tuned Model Weights	https://www.kaggle.com/models/rajarshi2712/rsicd-blip-image-caption-fine-tuned

This includes:

- the fine-tuned BLIP model (epoch 1-5 checkpoints)
- model metadata for deployment
