GIF Accessibility Reader: Local ML Rebuild & Model Comparison

Technical Report

Author: Jason Xu
Faculty Advisor: Professor Min Chen
Institution: University of Washington Bothell

1. Introduction

GIFs are one of the most common media formats shared on social platforms such as Twitter, Reddit, Discord, and messaging apps. While they provide rich expressive value, they remain largely inaccessible to visually impaired users because screen readers cannot inherently describe motion or visual context. Traditional accessibility tools may provide descriptions for static images, but animated content like GIFs requires specialized captioning strategies that capture not just the objects within a frame, but also motion, emotion, and temporal changes.

The original **GIF Accessibility Reader (2023)** attempted to address this problem by generating single-frame captions using a legacy codebase. However, it suffered from multiple issues: outdated dependencies, poor runtime performance, limited model support, and a fragile architecture that made further development difficult. Additionally, its single-frame captioning approach often failed to capture the dynamic elements that make GIFs meaningful.

This project rebuilds the entire system from the ground up, focusing on **modernizing the backend**, improving performance, simplifying model integration, and evaluating multiple machine learning models to determine an optimal captioning strategy for animated GIFs.

2. Project Objectives

The objectives of this capstone project were divided into four major areas:

2.1 Legacy System Analysis

The project began with a full audit of the legacy system from the 2023 research team. This included identifying:

- Unsupported or outdated Python modules
- Broken dependencies preventing local execution

- Inefficient or overly complex logic in both backend and Chrome extension
- Missing documentation and inconsistent code structures

This analysis provided the foundation for deciding which components to refactor, which to replace, and which to remove.

2.2 Backend Rebuild Using FastAPI

The old backend relied on a slow, synchronous architecture. It was replaced with a **new FastAPI backend**, which provides:

- Modern async support
- Clean API route design (e.g., /caption/blip/url)
- Better model lifecycle management
- Easier deployment and extension

2.3 Integration of Modern Machine Learning Models

The new system integrates several state-of-the-art vision models to study how different architectures perform on GIF captioning tasks:

- **BLIP** – fast, reliable image captioning baseline
- **MobileCLIP** – embedding-based classifier for label testing
- **BLIP-2** – multi-frame, image-to-text pipeline providing more expressive captions

This enables systematic evaluation of caption quality, runtime, and usefulness for accessibility tasks.

2.4 Model Benchmarking and Evaluation Pipeline

The system allows comparison of:

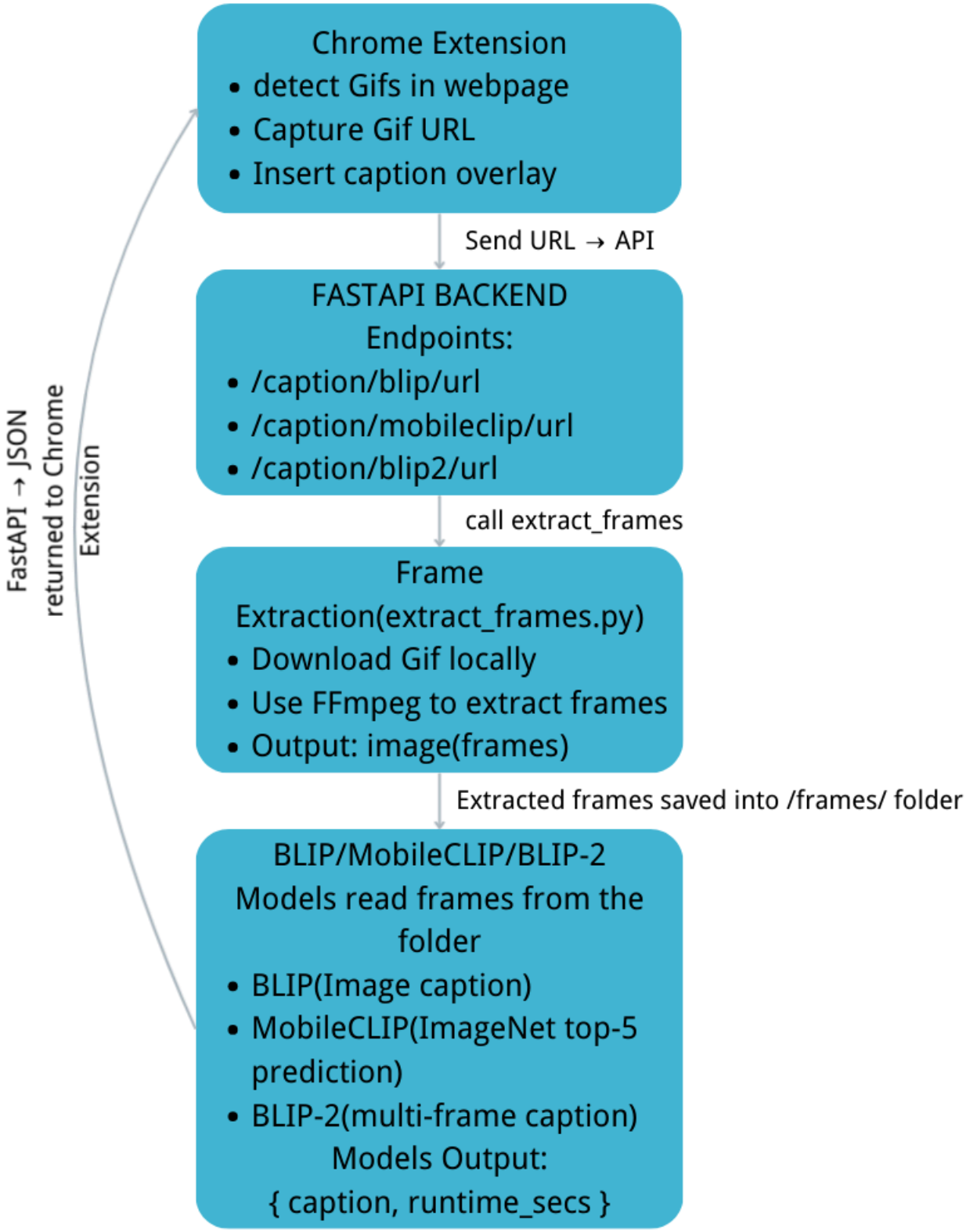
- **Single-frame vs multi-frame captioning**
- **Accuracy across emotion, object, text, and scene categories**
- **Runtime performance on CPU environments**

The evaluation goal was not only to identify the best model but to understand the trade-offs between speed and quality.

3. System Architecture

The system architecture for the modernized GIF Accessibility Reader consists of two major components—a Chrome Extension that interfaces with live web content and a FastAPI backend

responsible for caption generation. Together, these components form a lightweight, modular pipeline that processes GIFs and short video clips in real time. Throughout development, the architecture evolved significantly, especially regarding how machine learning models are deployed and accessed. The final design balances simplicity, extensibility, and compatibility with reused components from the 2023 project.



3.1 Chrome Extension

The Chrome Extension serves as the client-side detection and accessibility layer. It continuously monitors webpages using a MutationObserver, which allows it to detect newly inserted <img> and <video> elements on dynamic websites such as Twitter, Reddit, and Imgur. When new media elements appear, the extension inspects their source URLs to determine if they represent GIFs or short MP4 videos suitable for caption processing. Once a valid source is found, the extension sends a request to the backend to generate a descriptive caption.

Upon receiving the caption, the extension modifies the webpage in real time to improve accessibility. For GIFs, it injects an alt attribute so that screen readers can interpret the content. For videos, it applies an aria-label and enables keyboard focus using tabIndex. The extension also outlines processed media with a green border to provide immediate visual feedback. This workflow is entirely asynchronous, lightweight, and operates independently of the webpage itself, requiring only that the backend is running locally.

Importantly, the Chrome Extension does **not** allow users to switch between different captioning models. Although the project initially considered supporting multiple endpoints, the extension structure based on the legacy implementation was not designed for runtime model selection, and adding such functionality would require major architectural changes. Instead, the extension always calls a single unified backend endpoint, and the model used is determined solely by which backend configuration is running in the developer's VS Code terminal.

3.2 FastAPI Backend

The FastAPI backend provides all captioning services and encapsulates the machine learning model pipeline. Early in the project, the backend was designed with three distinct endpoints: /caption/blip/url, /caption/mobileclip/url, and /caption/blip2/url to allow clients to select a specific model. However, during integration with the reused Chrome Extension and the TestBench Web App, it became clear that these clients were not structured to support model selection and that implementing such functionality would require significant redesign.

Additionally, FastAPI can only run one model instance per server process. Launching the backend involves starting a specific application module:

uvicorn api_blip:app --reload

uvicorn api_mobileclip:app --reload

uvicorn api_blip2:app --reload

Since only one model configuration can be active at a time, providing three separate endpoints would be misleading; the frontend cannot dynamically switch models because the backend itself cannot host multiple models simultaneously.

For these reasons, the backend architecture was simplified to expose **one unified endpoint**:

/caption/blip/url

This endpoint name remains consistent regardless of which model is running internally. The chosen model is determined entirely by which backend module the developer launches. This unified endpoint dramatically simplifies integration, reduces failure points, and maintains full compatibility with the existing frontend components.

Internally, the backend handles all details of the inference workflow, including media downloading, GIF frame extraction, preprocessing, running the model, measuring inference runtime, and returning a structured JSON response. Because each backend module follows the same response format, both the Chrome Extension and the TestBench Web App can operate without modification, regardless of which model is being benchmarked.

3.3 ML Model Layer

The model layer consists of:

BLIP

- Fastest model
- Generates clear, simple sentences
- Works best for static or object-focused GIFs
- Ideal for real-time accessibility

MobileCLIP

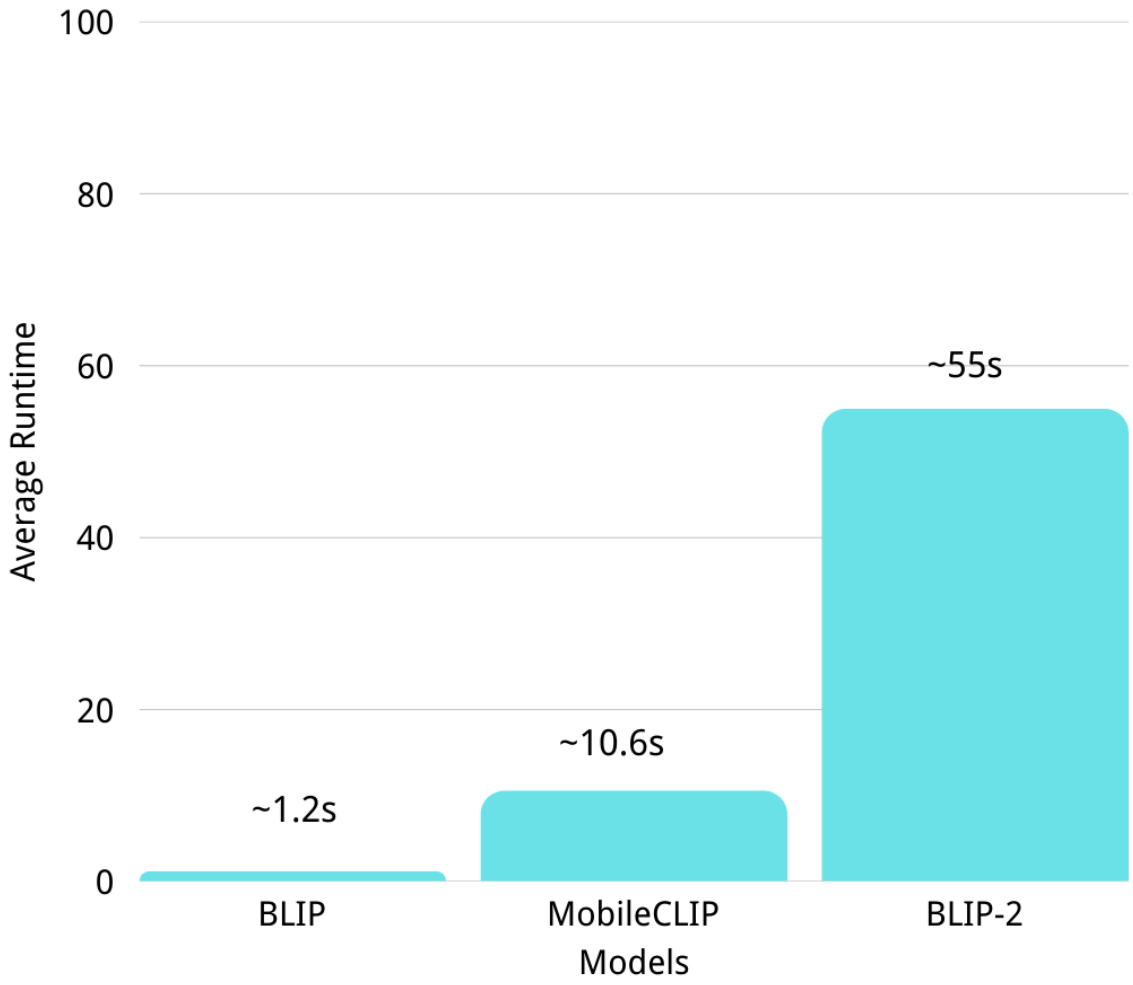
- Not a generative model
- Returns the nearest ImageNet label
- Not suitable for captioning, but useful as a baseline classifier

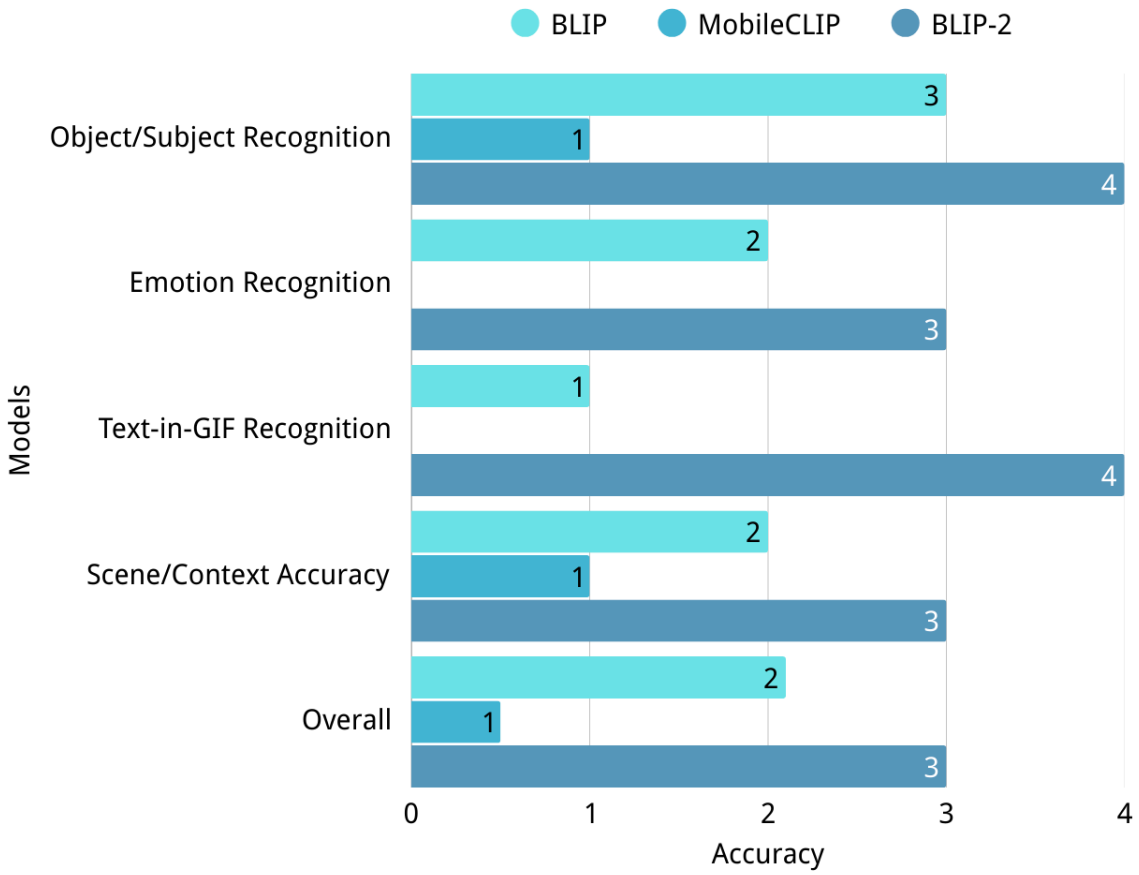
BLIP-2

- Highest-quality captions
- Better at emotions, intent, and multi-frame reasoning
- Very slow on CPU due to model size and cross-frame attention

The system uses BLIP as the default model for user-facing tools because it balances accuracy with real-time performance.

4. Model Evaluation


100
80
60
40
20
0
Average Runtime
~1.2s
~10.6s
~55s
BLIP
MobileCLIP
BLIP-2
Models



Model performance was measured across multiple categories:

- **Object & Subject Recognition**
- **Emotion Recognition**
- **Text-on-GIF detection**
- **Scene/Context Accuracy**
- **Overall Narrative Quality**

4.1 Evaluation Findings

- **BLIP**
  Fastest model, captures objects and general scenes well but misses emotional nuance and fails when text is embedded inside GIFs.

- **MobileCLIP**
  Produces only one-word labels. Often irrelevant or too general. Not suitable for accessibility purposes.

- **BLIP-2**
  Best caption accuracy overall. Produces rich descriptions and emotional context. However, runtime is significantly slower, up to 55 seconds per GIF on CPU.

5. Remaining Limitations

Despite improvements, several limitations remain:

5.1 Hardware Limitations

BLIP-2 requires GPU acceleration for practical real-time usage. CPU-only inference is too slow for integration into a live web accessibility tool.

5.2 Single-Frame vs Multi-Frame Limitations

GIFs are inherently multi-frame, but BLIP and MobileCLIP treat them as static images. This causes the model to miss:

- Motion direction
- Emotional progression
- Key context changes

5.3 Fast Motion Blur & Noise

GIFs often contain:

- Compression artifacts
- Fast transitions
- Inconsistent quality

These factors reduce model accuracy.

6. Future Work

Several enhancements can improve the system further:

6.1 Integrate Video-Caption Models

Models such as **VideoLLaMA**, **OpenFlamingo**, and **VILA** are specifically designed for multi-frame reasoning and could dramatically improve caption quality.

6.2 GPU-Accelerated Deployment

Running BLIP-2 or video models on cloud GPUs (e.g., Colab Pro, AWS, Modal) would solve runtime issues and allow multi-user access.

6.3 Public Web API Deployment

Packaging the FastAPI backend as a public service would allow:

- Third-party developers to integrate accessibility tools
- Browser extensions or mobile apps to consume the captions

7. Learning & Reflection

This project required combining skills across software engineering, machine learning, and web development. Key learning outcomes include:

7.1 ML Pipeline Integration

Learned how to load, optimize, and benchmark different models within a uniform API framework.

7.2 Rebuilding Legacy Systems

Gained experience evaluating inherited codebases, refactoring outdated modules, and modernizing architecture.

7.3 FastAPI Backend Design

Designed clean, extensible REST endpoints suitable for machine learning workloads.

7.4 Full-Stack Accessibility Tooling

Learned how browser extensions interact with backend APIs and how to inject accessibility features into live web pages.

7.5 Evaluation and Benchmarking Skills

Built evaluation scripts, measured runtime performance, and compared accuracy across model families.

8. Conclusion

This project successfully delivered a modernized, fully local GIF Accessibility Reader. The new FastAPI backend, multi-model evaluation pipeline, and rebuilt Chrome extension significantly improve the maintainability, performance, and future extensibility of the system.

The evaluation reveals that:

- **BLIP is the best model for real-time accessibility.**
- **BLIP-2 achieves the highest caption quality but requires GPU support.**
- **MobileCLIP is not suitable for caption generation.**

This work establishes a strong foundation for future research into multi-frame captioning models and real-world accessibility deployment.