Project Proposal: Image → Text → Sentiment Model Chaining and Inference Server Deployment

1. Introduction & Problem Statement

Modern machine learning applications often rely on multiple models working together in a pipeline rather than a single model performing an end-to-end task. This project aims to design, build, and deploy a **multi-model inference pipeline** where:

1. **Model 1** converts an input image into a descriptive caption (image-to-text).
2. **Model 2** analyzes the generated caption to determine its **sentiment** (positive, negative, or neutral).
3. The full pipeline is deployed behind a **unified inference server API**, containerized using Docker.

This allows users to submit an image and automatically receive both the caption and its corresponding sentiment in a single request. The goal is to explore how model chaining works in production environments, compare model performance across different architectures, and demonstrate best practices for serving composite AI systems.

The project addresses questions such as:

- How do chained models affect latency and error propagation?
- How accurately can image captions represent visual content for downstream NLP tasks?
- What infrastructure is needed to reliably serve multiple models through a single API?

---

2. Data Sources

Two datasets will be used—one for testing/evaluation and one for demonstrating the pipeline:

Image Captioning Evaluation Dataset

- **MS-COCO 2017 Validation Set**
  A standard dataset for evaluating image captioning models.
  Contains images with five human-generated captions each.
  Used to evaluate the accuracy of Model 1.

Sentiment Evaluation Dataset

- **SST-2 (Stanford Sentiment Treebank)**
  A common benchmark dataset for sentiment classification.
  Used to benchmark Model 2 independently.

User-Provided Images

To demonstrate the end-to-end pipeline, additional custom images (e.g., everyday objects, scenes) may be used.

---

3. High-Level Methods, Techniques & Technologies

Modeling Techniques

1. **Image → Text (Captioning Model)**
    - Options:
        - BLIP (HuggingFace)
        - ViT-GPT2 Image Captioning Model
        - OFA Vision-Language Model
    - Evaluated using BLEU, METEOR, and ROUGE metrics.
2. **Text → Sentiment (NLP Classifier)**
    - Options:
        - DistilBERT sentiment classifier

        - RoBERTa sentiment classifier
    - Evaluated using accuracy, F1 score, and confusion matrix.

Model Chaining Architecture

- Image → Caption → Sentiment pipeline
- Orchestration via Python backend, likely FastAPI or Flask
- Intermediate output sanitization and validation

Infrastructure & Deployment

- **Inference Server: FastAPI**
    - Endpoint: `/analyze-image`
    - Returns JSON {caption, sentiment, confidence}
- **Containerization: Docker**
    - Separate containers vs. unified container exploration
    - Docker Compose for multi-service deployment
- **Model Hosting**
    - Local HuggingFace inference
    - Optional GPU support

Evaluation

- Captioning accuracy (BLEU, METEOR)
- Sentiment accuracy (F1, accuracy)
- End-to-end latency and throughput

- Error propagation analysis across chained models

---

4. Products to Be Delivered

The final project deliverables will include:

1. Fully Functional Multi-Model Inference Server

- One API endpoint that:
    - Accepts an uploaded image
    - Generates a caption
    - Performs sentiment analysis on the caption
    - Returns both results as structured JSON
- Containerized using Docker (with a Dockerfile or Docker Compose)

2. Python Source Code

- Code for:
    - Image captioning module
    - Sentiment classification module
    - Pipeline orchestration logic
    - FastAPI server
- Clean, documented, with modular design.

3. Performance Evaluation Report

Contains:

- Model benchmarking (captioning + sentiment)
- End-to-end analysis
- Discussion of latency, bottlenecks, and improvement ideas
- Visualizations (confusion matrices, timing plots)

4. README Documentation

Includes:

- Setup instructions
- How to run the Docker container
- How to call the API (example curl commands / Python client)
- Example input/output format

5. Presentation or Demonstration Notebook

- Shows example images, generated captions, and corresponding sentiments
- Provides model comparison results
