CSC490 Assignment 2

Group 1 AGD-G
Avanti Tandon - 1010498695, Sark Asadourian - 1010726056,
Tianyu Luo - 1010130109, Tyson Caul - 1009951991

1 Aspirational Datasets

We ideally want a dataset of chart images paired with question-answer pairs where the answers are easy to judge (e.g., extrema, comparisons, and point lookups). If our adversarial diffusion attack can reliably flip answers on these simple questions, it is a strong indicator that it will also degrade performance on harder, multi-step reasoning questions.

In addition, we want charts to be readable and diverse in chart type, topic, origin, and visual style, with a large number of examples per chart family. The only metadata we require is a meaningful chart-type taxonomy label. Ideally, most charts would come from real-world sources and include the underlying data and the code to regenerate them (so we can re-render clean text and labels after perturbations). Additionally, a subset would include established adversarial baselines benchmarked on open-weight models for easy comparison.

2 Reality Check

Table 1: Open-source/Generated datasets for this project (links embedded)

Item	Description	Commentary
ChartQA-X	A chart question-answering dataset (28k charts) with explanations and reasoning. The charts are largely programmatically generated or synthetic, rather than primarily real-world figures.	Useful for our core requirement of easy-to-judge QA pairs over charts. However, it has limited real-world variety and style diversity. We will use it primarily for (i) the chart and question-answer component of our dataset, and (ii) extracting a coarse chart-type label when available.
ChartX	A smaller chart QA dataset with a broader variety of chart types and styles than ChartQA-X, with answers that are typically clear and gradable.	Complements ChartQA-X by increasing chart-type and style diversity, though it has fewer total examples. We can use it to test whether our attack generalizes to more a chart types.
ChartBench	A chart QA dataset emphasizing clean, diverse charts. Many examples omit in-chart labels and text, which reduces the amount of text the model can rely on.	Potentially a better fit for our attack setting. Unlabeled or low-text charts may let us focus the diffusion perturbation, while still keeping answers easy to verify.
ECharts Synthetic Generator (generated)	A script we could write that generates charts using Apache ECharts from random or templated data, and algorithmically generates QA pairs directly from the underlying data, along with a chart-type label.	This can scale chart volume and chart-type coverage, but it is not real-world data and requires significant engineering effort. Since many similar synthetic-chart generators already exist in prior work, it is not the best for this project.

3 Data-processing Pipelines

3.1 Overview

This section describes the end-to-end data pipeline used to (i) ingest chart QA datasets, (ii) normalize and store raw artifacts, (iii) generate adversarially perturbed charts via diffusion, and (iv) evaluate how model answers change under attack.

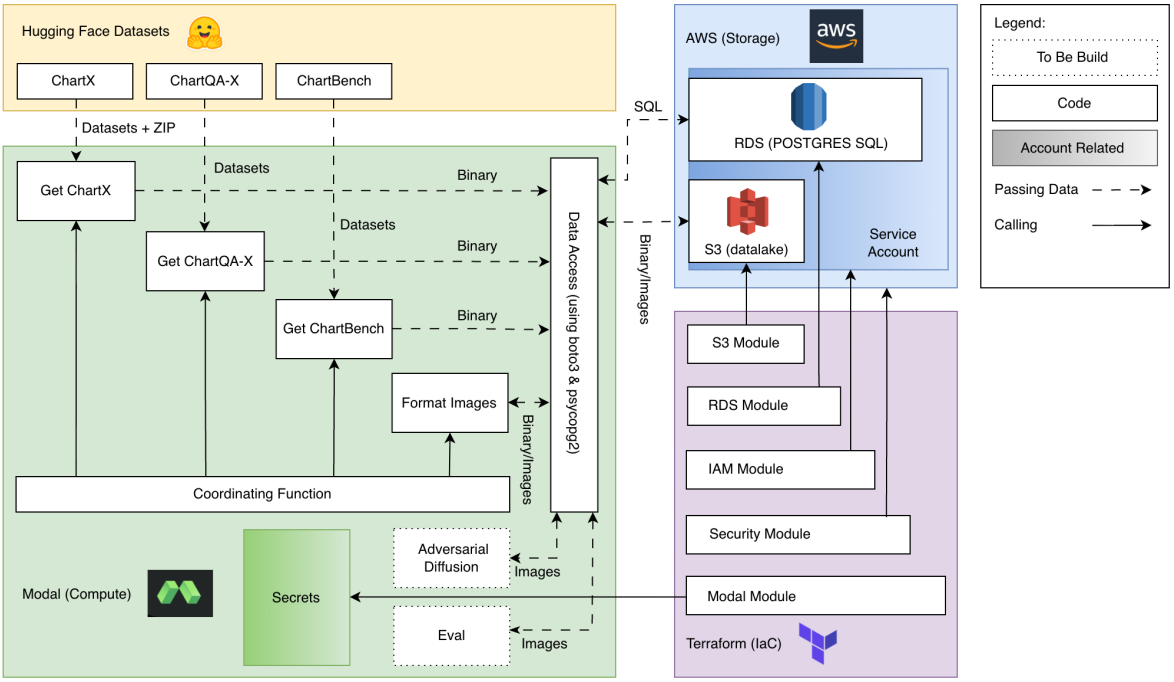

Figure 1: Data Pipeline Diagram.

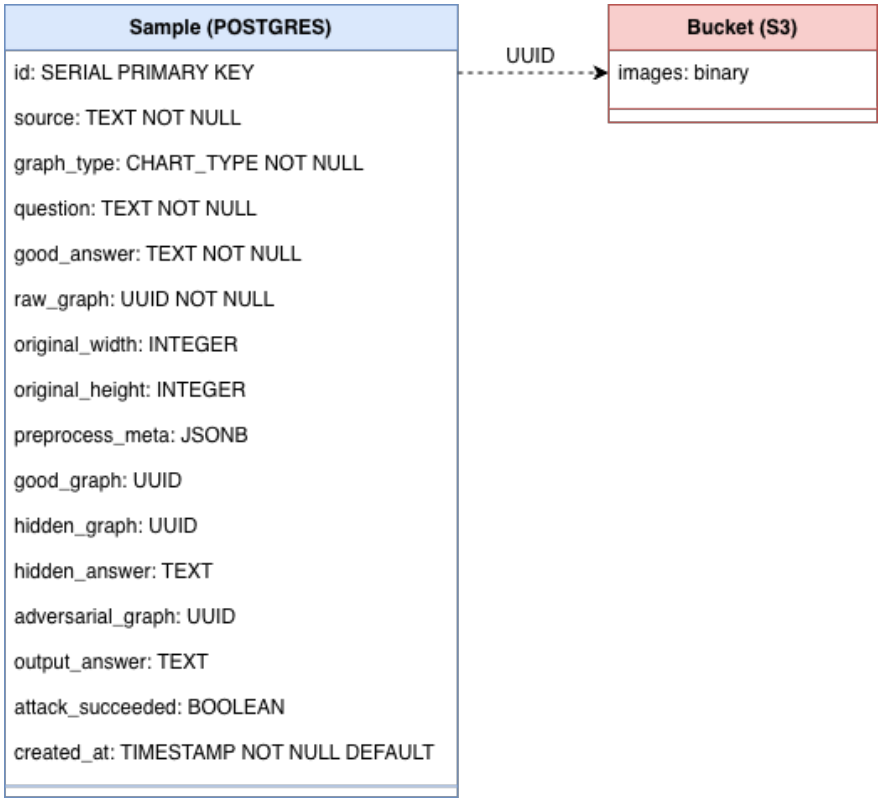

Figure 2: Data Schema Diagram.

3.2 Use cases:

- Initialize or refresh the PostgreSQL schema before experiments.
- Ingest chart QA data from Hugging Face using our data-access module (questions, answers, chart images, and chart-type labels when available).
- Validate that ingested data is correctly stored (S3 objects exist and metadata rows match) before running experiments.
- Generate adversarial chart images via diffusion and run the target model to produce answers on original vs. attacked charts.
- Store judge-LLM evaluations of answer correctness and attack effectiveness.

3.3 Future Development

Some features that will be implemented in the future:

- **End-to-end adversarial evaluation loop**: extend the pipeline so that each ingested chart is passed through an adversarial diffusion step, then evaluated by (i) a target vision-language model that answers the original questions, and (ii) a judge LLM that scores whether the attacked answer deviates from the ground truth and whether the perturbation remains visually plausible.
- **Text-safe perturbations (fallback)**: if end-to-end diffusion degrades label readability, restrict the adversarial diffusion to the visual primitives of the chart (marks, lines, bars, fills) while masking or cropping text regions so that axis labels and legends remain human-readable.

4 Infrastructure as Code Implementation

We use Terraform modules to provision the AWS infrastructure required by the pipeline. The configuration is split into clear, reusable components:

- **Security module**: defines the network boundary for the database (a PostgreSQL security group on port 5432 with explicitly configured allowed CIDR ranges).
- **RDS module**: provisions a PostgreSQL 15 instance and attaches the security group. It is used as the system-of-record for structured metadata (datasets, chart records, pipeline runs, and evaluation outputs).
- **S3 module**: provisions the data lake bucket used for raw and derived artifacts (original images, preprocessed images, and adversarial outputs).
- IAM module: Manages identity and access by provisioning the necessary roles and policies, ensuring least privilege access for the pipeline services interacting with AWS resources.
- Modal Secrets script: Executes a custom script within the Terraform workflow to securely add and manage Modal Secrets. This programmatic approach bridges the gap since there is currently no native Terraform provider or connection available for Modal.

The environment wiring exports the S3 bucket name and RDS endpoint as Terraform outputs so the pipeline can connect without manual configuration drift.

5 Disaster Recovery

Here is our **video**