Social Computing Lab at McGill University
Principal Investigator: Prof. Joseph Vybihal – https://www.cs.mcgill.ca/~jvybihal/index.php

---

McGill UNIVERSITY

Image Recognition & Ideology

Rayane Bouafia and Jules Lemée

Research Conducted from April 2024 to April 2025

Abstract

This project initially aimed to investigate algorithmic siloing and political bias on Instagram Reels by developing web scrapers to simulate user behavior and analyze the resulting content feed. Significant challenges with anti-scraping measures, account creation, and reliably influencing the algorithm led to difficulties in gathering conclusive data. A secondary component involved building a database of politically leaning Instagram accounts by scraping profiles, taking screenshots, and using AI to categorize them in ‘right-leaning’,‘left-leaning’, or neutral.

Due to persistent scraping obstacles, the project pivoted mid-way to focus on reproducing and potentially extending a 2019 research paper concerning the AI classification of political images. This involved setting up a legacy software environment (PyTorch 1.3.1 via Anaconda), accessing and preparing a large dataset on McGill's HPC cluster (Slurm), and adapting the original paper's code (ResNet50+SVC) to run within cluster time limits using checkpointing. We successfully executed the original paper's implementation (as of writing, the program is still running but no further obstacles are expected). Critically, due to time constraints and setup complexities, we were unable to proceed with the planned experiments using modern AI models (e.g., Vision Transformers, CLIP) for comparison.

Project Description

Purpose

The primary goals evolved over the project's duration. Initially, the purpose was to understand how user interactions influence the Instagram Reels algorithm, specifically concerning the prevalence and potential siloing of political content. A secondary goal was to develop automated methods for identifying politically leaning accounts. Following a pivot, the purpose shifted to reproducing baseline results from a 2019 paper on AI-based political image classification and setting the stage for evaluating newer computer vision models on the same task.

Description

The project encompassed two main phases:

1. Instagram Reels Analysis: This phase involved extensive research into recommendation systems and web scraping techniques. We attempted to build various bots using Selenium, Appium, Mitmproxy, and BeautifulSoup to simulate user actions (scrolling, liking, saving, following, searching hashtags) on Instagram Reels. The aim was to observe how different interactions affected the political leaning of the content presented by the algorithm. This included designing experiments to "warm up" accounts towards specific political leanings and developing a method to automatically classify accounts by screenshotting profiles and analyzing them with an external AI model (openai-4o-multimodal).
2. Political Image Classification AI: This phase focused on replicating experiments from the paper "Predicting Visual Political Bias using Webly Supervised Data and an Auxiliary Task". We obtained the associated dataset and aimed to first run the original authors' code (ResNet50 feature extractor + LinearSVC classifier) on the McGill Slurm GPU cluster. The longer-term goal, which was not reached, was to implement and compare the performance of more modern models like Vision Transformers (ViTs) and CLIP on the same classification task ('image-to-issue').

What We Built

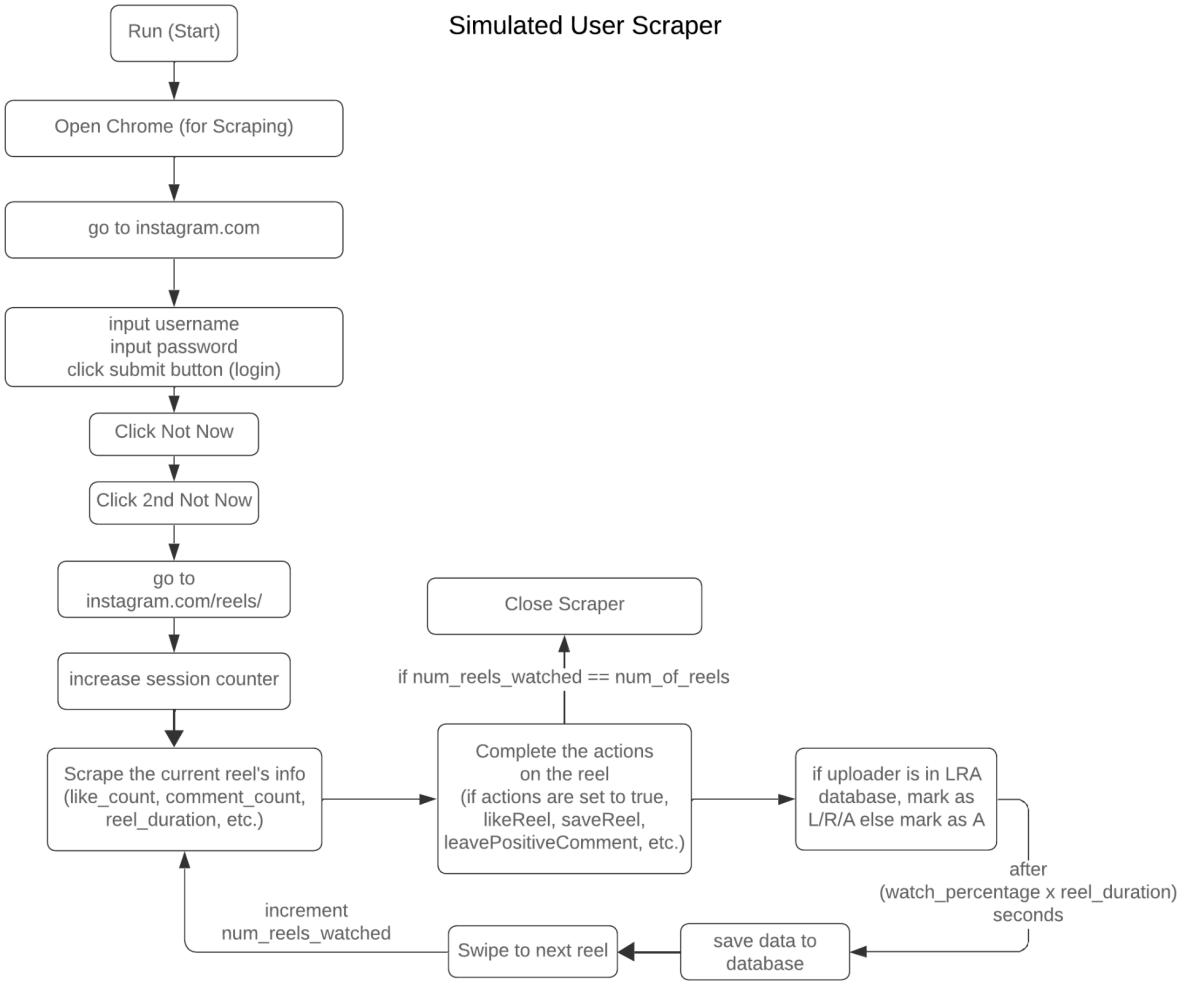
- Several prototype Instagram web scrapers using Selenium, Mitmproxy, Appium, and BeautifulSoup capable of logging in, navigating, scrolling Reels, liking, saving, and visiting profiles.
- A specific web scraping script designed to visit Instagram profiles, take screenshots, store them with AWS S3, query the OpenAI API to determine political leaning (Left/Right), and build a CSV database mapping usernames to predicted leanings.
- An Anaconda environment (`pytorch_1_3_1_env`) configured with legacy Python 3.7 and specific package versions from late 2019 to ensure compatibility with the target paper's code. Another environment was configured for the implementation of CLIP, with all required dependencies installed (not used).
- Rewrote Python scripts from the 2019 political image classification paper, modified to run on the McGill Slurm cluster, including the implementation of checkpointing to handle job time limits.
- Configuration files and submission scripts (`.sh`) for running jobs on the Slurm cluster.

What Our Software Does

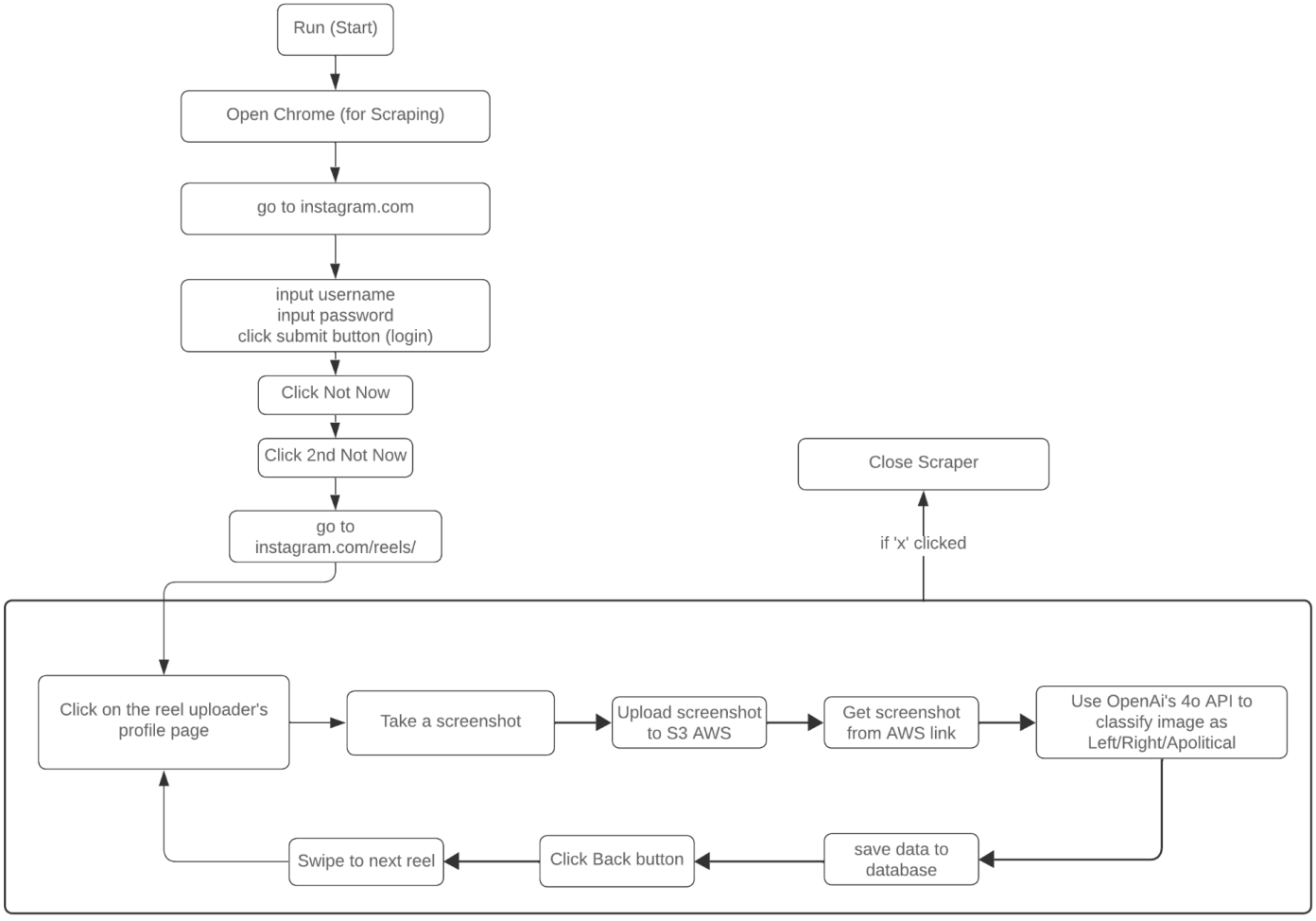
- The web scrapers attempt to automate interactions with Instagram to either study algorithmic responses or gather data on user profiles.
- The profile classification scraper automates the process of assigning a political leaning label to Instagram usernames based on their profile appearance via AI analysis.
- The Anaconda environment provides the necessary runtime conditions to execute the 2019 paper's code.
- The adapted AI scripts execute the image classification model training process (feature extraction and classifier training) on the designated dataset using the Slurm cluster's GPU resources.

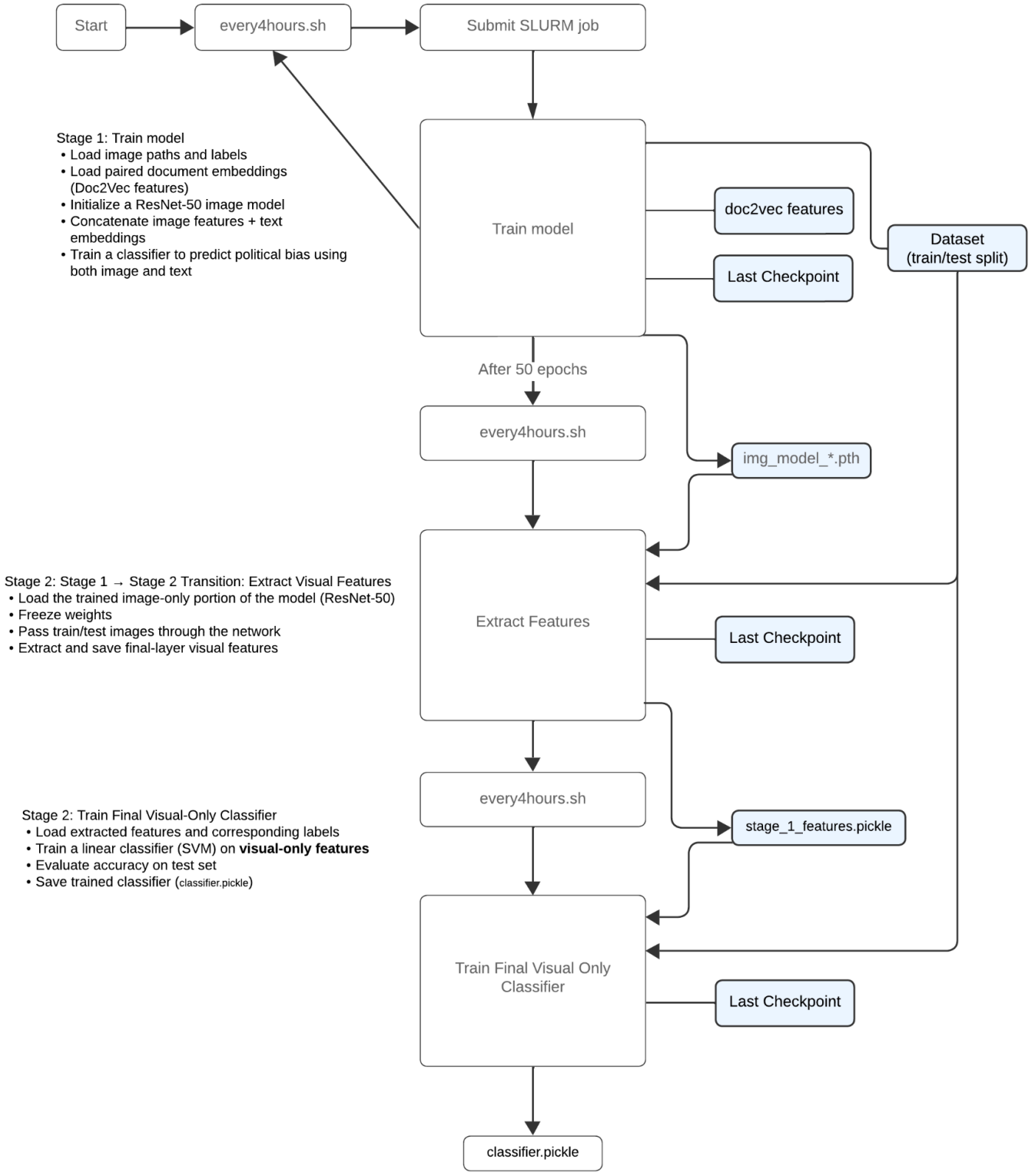
Project Diagrams

Storyboards



Database-Buulding Scraper.
The purpose of this scraper is to get a list of Instagram usernames alongside their political affiliation (left/right/apolitical). The LRA database will be used to determine a post's political standing by the simlated-user web scraper




Start
every4hours.sh
Submit SLURM job
Stage 1: Train model
• Load image paths and labels
• Load paired document embeddings (Doc2Vec features)
• Initialize a ResNet-50 image model
• Concatenate image features + text embeddings
• Train a classifier to predict political bias using both image and text
Train model
doc2vec features
Last Checkpoint
Dataset (train/test split)
After 50 epochs
every4hours.sh
img_model_*.pth
Stage 2: Stage 1 → Stage 2 Transition: Extract Visual Features
• Load the trained image-only portion of the model (ResNet-50)
• Freeze weights
• Pass train/test images through the network
• Extract and save final-layer visual features
Extract Features
Last Checkpoint
every4hours.sh
Stage 2: Train Final Visual-Only Classifier
• Load extracted features and corresponding labels
• Train a linear classifier (SVM) on visual-only features
• Evaluate accuracy on test set
• Save trained classifier (classifier.pickle)
stage_1_features.pickle
Train Final Visual Only Classifier
Last Checkpoint
classifier.pickle

Domain Models

Domain Model A — *Instagram Reels Political-Bias Scraping & Simulation*

1. Domain Overview

This subsystem automates Instagram-web sessions to observe how different user actions (like, follow, save, etc.) bias the Reels recommendation feed toward political content. It simulates human behaviour with Selenium, records every reel encountered, and stores structured observations for later analysis of algorithmic siloing.

2. Key Domain Concepts (Entities)

Entity	Type	Fields / Properties	Constraints
UserAccount	JSON (persisted to a secrets vault)	username :str, password :str, created_at :datetime, status :{Active, Banned, Flagged}, seed_leaning :{None, Left, Right}	username unique; password encrypted at rest
ScrapingSession	CSV row (sessions.csv)	session_id :int, account_username :str, start_ts :datetime, end_ts :datetime, condition :int (1 = unconditional, 2 = only-political-uploader), quit_after :int (#reels)	end_ts > start_ts
ReelObservation	CSV row (data_output.csv)	id :int, session_id :int, url :str, uploader :str, caption :str, like_cnt :int, comment_cnt :int, duration_s :float, watch_time_s :float, watch_pct :float, liked :bool, saved :bool, followed :bool, visited_profile :bool, shared :bool, positive_comment :bool, negative_comment :bool, not_interested :bool, logged_at :datetime	(watch_time_s ≤ duration_s); watch_pct = watch_time_s/duration_s
UploaderProfile	row in **PoliticalUserDB** (CSV)	username :str, predicted_leaning :{Left, Right, Unknown}, followers :int, sampled_on :datetime	username unique
ProxyEndpoint	YAML entry	ip :str, port :int, type :{Residential, Datacenter}, status :{Healthy, Banned}	rotate after n requests
Screenshot	PNG file + JSON side-car	path :str, account_username :str, taken_at :datetime, classification_id :str	

3. Relationships Between Entities

- A **UserAccount** *initiates* **many* ScrapingSessions (1-to-many).
- Each **ScrapingSession** *produces* **many* ReelObservations; a ReelObservation *belongs to* exactly one session (1-to-many).
- A **ReelObservation** *references* exactly one **UploaderProfile** (many-to-1).
  A **UserAccount** *may follow* many **UploaderProfiles**; an UploaderProfile *may be followed by* many UserAccounts (many-to-many).
- Each **Screenshot** *documents* one UploaderProfile, but an UploaderProfile can have many Screenshots over time (1-to-many).
- ScrapingSessions *reuse* multiple **ProxyEndpoints** during execution (many-to-many, time-boxed rotation).

4. Domain Operations / Processes

Operation	Inputs	Outputs	Preconditions / Invariants
open_reels	username, password	Live browser session at *instagram.com/reels/*	Credentials valid; proxy assigned; CAPTCHA not triggered
scrape	(see code) plus behavioural flags (liked, saved, …)	Appended rows in data_output.csv	Browser at Reels feed; session timer < quit_after
click_like / click_save / click_follow	current_reel WebElement	DOM mutated, action logged	Button present & enabled; Instagram not rate-limiting
get_reel_duration	current_reel	duration_s	Video tag present
classify_profile_screenshot	Screenshot PNG	predicted_leaning label	Screenshot uploaded; AI service reachable
rotate_proxy	None (cron)	new ProxyEndpoint bound to driver	At least one healthy proxy in pool

5. External Systems / Interfaces

- **Instagram Web UI** – primary data source & action target.
- **OpenAI API** – infers political leaning from profile screenshots.
- **AWS S3** – persistent store for screenshots and CSV logs
- **GitLab & Github Repo** – version control for scrapers and data-schema definitions.

Domain Model B — *Political Image-to-Issue Classification Re-implementation*

1. Domain Overview

This subsystem reproduces—and later extends—the 2019 IJCV study that predicts the political "issue" depicted in an image. It ingests a 1 M-image dataset, extracts CNN/ViT features, trains a classifier (Linear-SVC in the baseline), and logs metrics for comparison with newer vision models.

2. Key Domain Concepts (Entities)

Entity	Type	Fields / Properties	Constraints
RawImage	File on cluster filesystem	path :str, sha256 :str, split :{train,val,test}, issue_label :str (≈ 19 issues), leaning_label :{Left,Right,Neutral}, width,height :int	sha256 unique; file must exist
FeatureVector	NPY row (numpy array)	image_sha256 :str (FK), model_name :str, layer :str, vector :float, extracted_at :datetime	composite PK (image_sha256,model_name,layer)
ModelDefinition	JSON	model_name :str, arch :{ResNet50,ViT-B/16,CLIP-ViT-L}, pretrained_on :str, feature_dim :int, version :str	model_name unique
TrainingRun	CSV row (runs.csv)	run_id :int, model_name :str (FK), classifier_type :{LinearSVC,MLP,PromptTune}, started_at,ended_at :datetime, status :{Running,Finished,Failed}, accuracy :float, macro_f1 :float, slurm_job_id :str	ended_at ≥ started_at; metrics nullable until completion
Checkpoint	File (*.pt or *.pkl)	run_id :int (FK), epoch :int, path :str, saved_at :datetime	one latest checkpoint flagged is_best :bool
IssueCategory	YAML seed file	name :str, description :str, sample_count :int	fixed, 19 categories
GPUResource	Slurm queue record	job_id :str, gpu_type :str, node :str, allocated_hours :int	one GPUResource per TrainingRun

3. Relationships Between Entities

- Each **RawImage** *belongs to* exactly one **IssueCategory**; a category has many images (1-to-many).
- A **RawImage** *produces* one **FeatureVector per** ModelDefinition–Layer pair (1-to-many).
- A **ModelDefinition** *is used by* many TrainingRuns (1-to-many).
- A **TrainingRun** *writes* many **Checkpoints**; the run references exactly one GPUResource (1-to-1).
- FeatureVectors from many images *feed* into one TrainingRun (many-to-1).

4. Domain Operations / Processes

Operation	Inputs	Outputs	Preconditions
extract_features	model_name, image_batch	.npy FeatureVector rows	Model weights cached; batch fits GPU memory
train_classifier	feature_matrix, labels, classifier_type	Trained weights (*.pkl)	Features & labels aligned; class imbalance handled
evaluate_model	trained_classifier, test_features	accuracy, macro_f1, confusion matrix	No missing labels; checkpoint hash matches run
save_checkpoint	run_state, epoch	Checkpoint file	Disk quota available
resume_from_checkpoint	checkpoint_path	Restored run state	Checkpoint file exists & checksum verified
submit_slurm_job	Shell script, run_id	job_id	Cluster quota not exceeded; environment module loaded
compare_models_report	List[TrainingRun]	Markdown/CSV summary	All runs finished; metrics populated

5. External Systems / Interfaces

- **McGill Slurm GPU Cluster** – schedules extract_features & train_classifier jobs.
- **Anaconda Environment pytorch_1_3_1_env** – frozen legacy dependencies for baseline code.
- **Cluster File System on mimi (/home/2022/rbouaf)** – stores raw images, features, checkpoints.
- **GitLab & Github Repository** – version control & CI for training scripts.
- **Vision-Transformer / CLIP model hubs** – download modern pretrained weights for extension experiments.

Experiments

Descriptions

Instagram Political Feed Observations

- Goal: Determine what actions (like, follow, save) taken on a political post with a fresh Instagram account could bias the Reels feed towards similarly political content, and if these biases differed across the political spectrum.
- Setup: Create a new Instagram account (manually on mobile). Turn off content filters (‘Don’t limit’ applied to political content, found in settings->‘More’->sensitive content)
- Procedure: Open the Reels feed on the account using Selenium scraper. Scroll through the feed, recording the content of each reel encountered with our software, and once political content was identified, the program would record the kind, and, depending on the experiment, take an action on that observation.
- Measurement: The CSV database would record the session number and observation features, and we’d measure if an action on a type of political piece of content would affect the frequency that the same type of political content would reappear later.

Instagram Political Feed Priming

- Goal: Determine if explicitly interacting with right-wing political content on a fresh Instagram account quickly biases the Reels feed towards similar content (manually, to get an account to start showing us political content in general).
- Setup: Create a new Instagram account (manually on mobile). Turn off content filters (‘Don’t limit’ applied to political content, found in settings->‘More’->sensitive content). Follow specific right-wing figures (e.g., Ben Shapiro, Donald Trump). Explore related hashtags (e.g., #conservative, #maga). Spend approx. 10 minutes ‘warming up’ the account.
- Procedure: Open the Reels feed on the account (tested on both mobile and desktop browser). Scroll through the feed, recording the content of each reel encountered (specifically noting political relevance).
- Measurement: Count the number of scrolls required to encounter political reels, and the density of political reels within a short session (e.g., first 20 reels).

Reproducing 2019 Political Image Classification Baseline

- Goal: Replicate the image-to-issue classification performance reported in the reference 2019 paper using their specified methodology (ResNet50 features + LinearSVC) and dataset.
- Setup: Access the McGill Slurm GPU cluster. Activate the pre-configured legacy Anaconda environment (pytorch_1_3_1_env). Ensure the dataset is accessible at the correct path. Use the adapted training scripts incorporating checkpointing.
- Procedure: Submit the Slurm job to run the training script. The script should load the dataset, extract features using the ResNet50 model (pre-trained on ImageNet), train a LinearSVC classifier on these features, and (ideally) evaluate performance on a test set.

- Measurement: Monitor Slurm logs for successful execution or errors. (Intended: Record final classification accuracy/F1 scores and compare to the paper's reported results).

Results

Experiment 1 (Instagram Observations):
Our automated experiment results never saw the light of day, as anti-bot obstructed our bots from simply being able to view political content, point blank. This led us to design a second experiment, orchestrated manually, to test a workaround.

Experiment 2 (Instagram Priming):
The manual experiments demonstrated that it is possible to rapidly bias the Reels feed. After minimal targeted interaction (following 3 accounts, exploring hashtags), political content appeared very quickly. One session reported the first reel being political (Trump). Another reported getting "20 republican reels in a row" with "nothing else," indicating strong biasing is achievable manually. However, automating this with scrapers proved extremely difficult due to bans and detection. Furthermore, even the manual method failed quickly, we assume because our devices were detected behaving unnaturally, and once flagged, we found no further workarounds.

Experiment 3 (AI Reproduction):
The setup phase was successful: the legacy Anaconda environment was created, dependencies installed, dataset accessed, and code adapted for Slurm with checkpointing. The training script was successfully launched and executed on the Slurm cluster. The experiment successfully demonstrated the feasibility of running the code in the target environment but did not achieve the goal of replicating the paper's results due to runtime issues and time constraints. As of the date of this writing, the first of three steps in the author's code was successful (the initial model training on the 1 million file dataset), but the second step, extraction of features, was still in progress.

Reflection

What We Would Do Differently

Underestimated Scraping Difficulty: We significantly underestimated the technical challenges and time investment required to overcome Instagram's sophisticated anti-scraping measures. We spent too long trying to make fragile scrapers work.
Proxy Infrastructure: We didn't secure or implement a robust proxy management system early on, which was critical for avoiding IP bans and essential for any scraping at scale.
Pivot Timing: Perhaps we should have pivoted away from the Instagram Reels algorithm analysis earlier once the depth of the scraping challenges became apparent.
Legacy Code Reproduction: We underestimated the difficulty of setting up and running code from 2019 due to dependency conflicts ("dependency hell"). More time should have been allocated for environment setup and debugging the baseline model before planning extensions.

Cloud Costs: Should have monitored cloud usage more closely or relied more heavily on university resources from the start to avoid unexpected costs.

What We Would Do Again

Adaptability/Pivoting: Recognizing the scraping roadblock and pivoting to the AI paper reproduction was a good strategic decision to ensure some tangible progress could be made.
Problem Decomposition: Breaking down the AI task into steps (get access, get data, build environment, run original code, run new code) was the correct approach.
Leveraging University Resources: Successfully navigating the process to gain access to and utilize the McGill HPC (Slurm) cluster and storage was a significant and necessary achievement.
Creative Problem Solving: The idea to use profile screenshots + OpenAI for classification was an innovative workaround for directly scraping post content or structured bio data.
Persistence: The team persisted through numerous technical and administrative challenges (GPU access, storage limits, dependency issues, scraping bans).

Key References

Core AI Political Image Classification Papers & Resources

- Thomas, C. & Kovashka, A. (2019). *Predicting Visual Political Bias using Webly Supervised Data* Retrieved from https://people.cs.pitt.edu/~chris/politics/paper.pdf, longer form publication available here: https://people.cs.pitt.edu/~kovashka/thomas_kovashka_politics_ijcv2021_preprint.pdf
- Vybihal, J., & Deblanc, M. (2022). Analysis of the Impact of Algorithms on Siloing Users: Special Focus on YouTube. In *AI & Society: Tensions and Opportunities*. Taylor & Francis. Retrieved from ResearchGate: https://www.researchgate.net/publication/364557056_Analysis_of_the_Impact_of_Algorithms_on_Siloing_Users_Special_Focus_on_YouTube (Also listed via publisher link: https://www.taylorfrancis.com/chapters/edit/10.4324/9781003314882-10/analysis-impact-algorithms-siloing-users-special-focus-youtube-joseph-vybihal-mika-joseph-paul-deblanc)
- OpenAI. (n.d.). *CLIP: Connecting Text and Images*. Retrieved from https://openai.com/index/clip/
- Robillard, M. P., Maalej, W., Walker, R. J., & Zimmermann, T. (Eds.). (2014). *Recommendation Systems in Software Engineering*. Springer Science and Business. Retrieved from https://link.springer.com/book/10.1007/978-3-642-45135-5
- De, S., Dey, S., Bhatia, S., Bhattacharyya, S. (2022). An Introduction to Data Mining in Social Networks. In L. C. Jain, H. H. Nguyen, & S. S. Howlett (Eds.), *Advanced Data Mining Tools and Methods for Social Computing* (pp. 1-23). Elsevier. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/B9780323857086000084