CSC490 Assignment 2: Data processing Pipelines and Infrastructure as code

Sohee Goo / 1008065479 Akshaya Deepak Ramachandran / 1008806810
Maryam Taj / 1008990362 Kashish Mittal / 1009115315

1 Part One: Aspirational Datasets

Ideally, we would have access to a dataset containing a curated collection of high resolution images of authenticated artworks spanning multiple artists, periods, and styles, paired with verified human-made forgeries of those same works or artists. Such a dataset would allow the model to learn distinctive artistic signatures while also capturing the subtle deviations introduced by forgers.

In addition, datasets containing artist-level and artwork-level metadata, such as provenance records, creation dates, materials, and known forget techniques, would support the construction of a knowledge base for the RAG system, enabling more grounded and interpretable explanations of the model's predictions.

Data Schemas:

Table 1: **User**: Represents a registered user, storing authentication credentials and historical interaction data related to artwork authentication.

Field	Type	Description
user_id	Int	Unique identifier for each user
username	String	Unique username chosen by the user for login
password	String	Hashed password for authentication
email	String	User's registered email address for account verification and login

Table 2: **Results**: Represents the output generated by the system for a user submitted artwork, including the model's authenticity prediction and its accompanying explanation.

Field	Type	Description
results_id	Int	Unique identifier for each authentication result
artwork_image	String	Link to the submitted artwork image
artwork_name	String?	Title of the artwork as provided by the user
forgery_score	Float	Model-generated probability score indicating likelihood of forgery on scale from 0 to 1
explanation	String	Explanation generated by the RAG system, grounded in retrieved artist and artwork metadata
user_id	Int?	Foreign Key, Refers to the user that generated this result

Table 3: **Artwork**: Represents an artwork sample in the dataset, containing image data and associated metadata used to train and evaluate models for art forgery detection.

Field	Type	Description
image	String	Link or path to the artwork image
artwork_name	String	Title or name of the artwork
artist	String	Attributed artist of the artwork
capture_method	String	Method used to digitize the artwork (eg. photograph, scan, etc)
forgery_label	String	Ground-truth authenticity label indicating whether the artwork is authentic or forged

Table 4: **Artist**: Represents an artist entity within the dataset, capturing stylistic and contextual information used to support interpretation and explanation of artwork authenticity.

Field	**Type**	**Description**
artist_name	String	Full name of the artist
style_information	String	Description of the artist's characteristic style, techniques, and associated artistic movements
known_forgery_techniques	String	Documented forgery techniques commonly associated with the artist or style

Table 5: **ArtworkInformation**: Represents detailed, artwork-specific metadata providing historical, material, and provenance context for an individual piece.

Field	**Type**	**Description**
artwork_name	String	Title or name of the artwork
artist	String	Attributed artist of the artwork
creation_date	DateTime	Date or estimated period when the artwork was created
materials	String	Materials or mediums used in the artwork
artwork_style	String	Artistic style or movement associated with the artwork
provenance_records	String	Summary of known ownership records and provenance information

2 Part Two: Reality Check

2.1 Data for training the Computer Vision Model

Data Set	Description	Commentary
Vincent van Gogh Authentication Dataset [1].	A dataset centered on Vincent Van Gogh's work, consisting of two components: (1) The VGDB-2016 Dataset with 126 original artworks from Vincent van Gogh and (2) a contrast dataset containing stylistically similar works by contemporary artists, non-autograph copies, explicitly labeled human-made forgeries (e.g., by Otto Wacker and John Myatt) and synthetic fakes using Stable Diffusion 2.1 and StyleGAN3.	This dataset is particularly helpful for our project due to its inclusion of confirmed human made forgeries, which is rare in publicly available art datasets. Prior work demonstrates that incorporating synthetic images alongside authentic and forged artworks can improve a model's ability to recognize human-made forgeries, and thus the synthetic fakes provided by this dataset would also be beneficial for our project. A limitation is that this dataset focuses on a single artist, which may restrict generalization to broader artistic domains and necessitates complementary datasets for multi-artist evaluation.
Rijksmuseum Data Services [2].	The Rijksmuseum provides open access to structured metadata and high-resolution images from the Rijksmuseum's collection. Images are delivered via the IIIF Image API.	This dataset provides high-quality, expertly curated reference images that are valuable for learning fine-grained visual features and material characteristics relevant to art authentication. Its use in prior art authentication research supports its reliability and relevance for our project. For our project, it serves as a source of high-fidelity authentic artworks and rich metadata that can be incorporated into the RAG knowledge base to support interpretable model explanations. A limitation is that the dataset does not explicitly label forged artworks, requiring it to be combined with external forgery or contrast datasets for supervised forgery detection.
Wikimedia Commons [3].	A publicly accessible media repository containing digital images of artworks, including works created by historically documented art forgers such as Han van Meegeren and Wolfgang Beltracchi. Images were retrieved using the MediaWiki API based on category membership and associated file metadata. Each entry includes descriptive metadata such as title, artist attribution, category tags, licensing information, and source URL. All images were obtained under public domain or Creative Commons licenses at the time of collection.	This dataset provides access to documented human-made forgeries across multiple artists and stylistic periods, helping to mitigate the single-artist limitation of the Van Gogh dataset. It enables broader evaluation of forgery detection models in more diverse artistic contexts and supports supervised training with historically grounded examples. A limitation is that Wikimedia Commons is community-curated, meaning labels may require external verification to ensure historical accuracy. The dataset may also exhibit class imbalance and variability in image resolution and quality.

2.2 Data for training the RAG Pipeline

Data Set	**Description**	**Commentary**
The Metropolitan Museum of Art Open Access [4].	An open-access metadata collection provided by The Metropolitan Museum of Art (The Met), containing structured catalog information for more than 420,000 artworks in the museum's public domain collection. From this dataset, we extract both artist-level and artwork-level information to support contextual analysis. At the artist level, the metadata captures details about the creator's identity, professional role, biographical background, nationality, lifespan, and demographic context. At the artwork level, the metadata describes the object's title and type, cultural and historical context, temporal placement, material composition, physical characteristics, provenance and credit information, and geographic associations. Together, this structured metadata supplies comprehensive historical, stylistic, temporal, geographic, and material context for each artwork, enabling richer interpretation and analysis.	This dataset serves as a high-confidence source of institutionally curated, authentic artwork metadata. Within our RAG pipeline, the metadata is used to retrieve contextually similar works and provide interpretable explanations grounded in verified historical information. Specifically, fields such as medium, period, cultural origin, artist lifespan, and geographic context support reasoning about stylistic consistency and historical plausibility in authentication tasks. A limitation is that the dataset provides structured catalog metadata rather than technical conservation data (e.g., pigment analysis, or material composition studies), which may restrict the depth of material-level authentication explanations.
WikiData [5].	WikiArt is a publicly accessible online art database that provides curated biographical and contextual information about artists from a wide range of periods, regions, and artistic movements. The data we scraped consists of structured textual descriptions focused on individual artists. The collected metadata includes biographical details such as birth and death information, nationality, and historical context, along with descriptions of the artist's stylistic characteristics, associated movements, thematic tendencies, and overall artistic significance. This information situates each artist within broader art historical developments and highlights defining elements of their practice. Overall, the dataset provides rich stylistic, temporal, and contextual grounding at the artist level, supporting tasks such as attribution analysis, stylistic comparison, and knowledge augmentation within the system.	One major advantage of this dataset is that it allows us to ensure coverage of specific artists included in our training and testing sets. Because the data is collected at the artist level, we can deliberately gather contextual and stylistic information for each artist represented in our model. This guarantees alignment between our metadata and the artworks being evaluated, strengthening interpretability and supporting more informed attribution or stylistic analysis. However, a key limitation is that the data collection process requires manual scraping for each individual artist of interest. This makes the pipeline less scalable and more time-intensive, especially as the number of artists increases. Unlike bulk-downloadable datasets, expanding coverage requires repeated targeted extraction, which introduces additional engineering overhead and potential inconsistencies across entries.

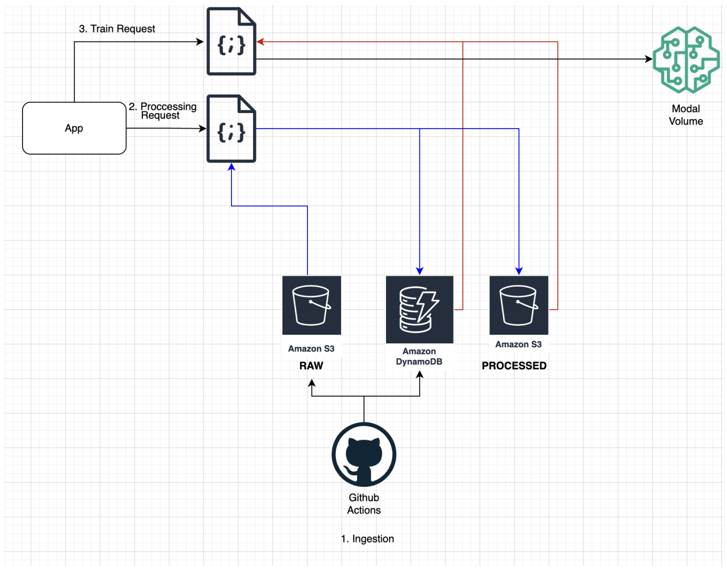

Figure 1: Training Data Pipeline Diagram

3 Part Three: Data-processing Pipelines

3.1 Training Pipeline

Data Schemas: Refer to schemas file.

Pipeline Diagram:

Pipeline Description:

- Ingestion: GitHub Actions triggers a script, which uploads local data to S3 bucket for storing raw images during training, and populates DynamoDB with their metadata.
- Cleaning: Each image is converted to RGB, and encoded as JPEG. Additionally, images that are too small are removed.
- Transformation: Each image is divided into 2^p by 2^p equally sized patches, with p depending on the resolution of the original image as follows: p = 2, if the smaller side of an image is larger than 1024 pixels, and p = 1, if the smaller side is larger than 512 pixels and smaller than 1024. We also include a center-cropped square from the full image. All of these resulting sub-images are normalized to 256×256 pixels. This patching method allows the models to extract very fine-grained brushstroke level information from the smaller patches, but also more compositional and representational features from the larger patches.
- Data lake/warehouse design: Each patch is stored in S3 bucket for storing processed images during training, and its metadata is uploaded to DynamoDB.
- Using stratified k-fold cross validation, the patches are split into train and test folds (i.e., this ensures the same ratio of original:forgery:imitation in train, and test sets, so they are representative samples). Within the train set, the patches are further divided into train and validation set. We train the Swin transformer on the train set, finetune its hyperparameters using the validation set, and evaluate per-fold performance on the test set. The optimal hyperparameters, and the entire training run's metadata is stored in DynamoDB, while the optimal model weights are stored in a Modal volume.

3.2 RAG Data-processing Pipeline

Data Schemas:

1. MET Open Access Data Schema:

Artwork Title: {Title}
Object Type: {Object Name}
Classification: {Classification}

Artist: {Artist Display Name}
Nationality: {Artist Nationality}

Lifespan: {Begin Date}–{End Date}

Cultural Context: {Culture}
Period: {Period}
Date Range: {Object Begin Date}–{Object End Date}

Medium: {Medium}
Dimensions: {Dimensions}
Credit Line: {Credit Line}

2. WikiArt Data Schema:

Artist: {artistLabel}
Description: {description}

Born: {birth}
Died: {death}
Citizenship: {citizenship}

Movements: {movements}
Genres: {genres}
Occupations: {occupations}
Fields: {fields}

Influenced By: {influencedBy}
Notable Works: {notableWorks}

Pipeline Diagram:

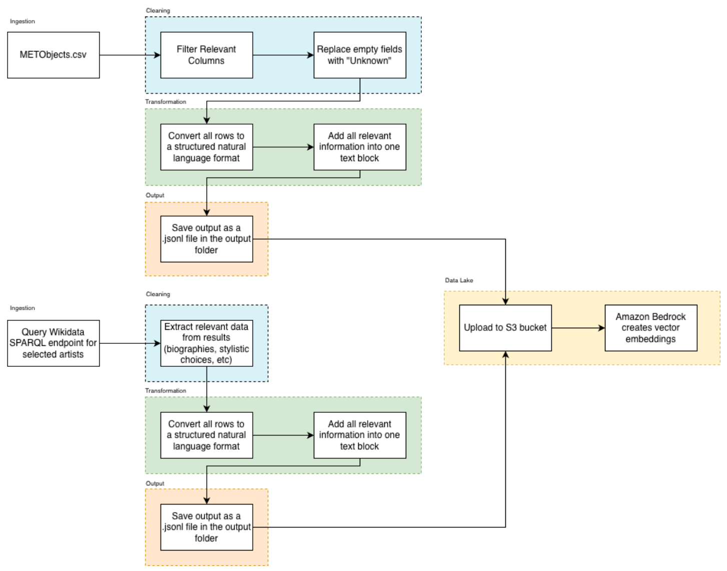
Ingestion
METObjects.csv
Cleaning
Filter Relevant Columns
Replace empty fields with "Unknown"
Transformation
Convert all rows to a structured natural language format
Add all relevant information into one text block
Output
Save output as a .jsonl file in the output folder
Ingestion
Query Wikidata SPARQL endpoint for selected artists
Cleaning
Extract relevant data from results (biographies, stylistic choices, etc)
Transformation
Convert all rows to a structured natural language format
Add all relevant information into one text block
Output
Save output as a .jsonl file in the output folder
Data Lake
Upload to S3 bucket
Amazon Bedrock creates vector embeddings

Figure 2: RAG Data Pipeline Diagram

Pipeline Description:

MET Data Pipeline: This pipeline will run once to populate our S3 bucket with the relevant data.

- Ingestion: The MET Open Access CSV file containing structured artwork data is loaded into the pipeline. All fields are read as strings to prevent type inconsistencies and ensure uniform handling of data.
- Cleaning: Only the artist-level and artwork-level metadata relevant for attribution and contextual explanation are retained. Empty strings and null values are replaced with “Unknown” to prevent embedding noise and avoid the appearance of missing or invalid data. Highly sparse or unused attributes are removed to streamline the process.
- Transformation: Each artwork entry is converted into a structured natural language document formatted for semantic embedding. Artist and artwork metadata are combined into a single contextual text block, ensuring that each record contains a rich description suitable for embedding models.

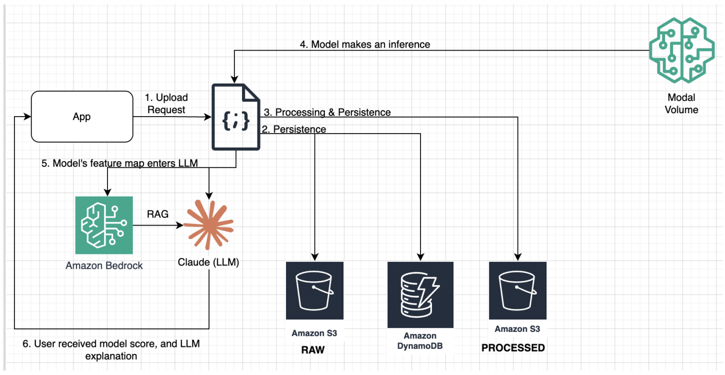
4. Model makes an inference
Modal Volume
App
1. Upload Request
3. Processing & Persistence
2. Persistence
5. Model's feature map enters LLM
RAG
Amazon Bedrock
Claude (LLM)
Amazon S3
RAW
Amazon DynamoDB
Amazon S3
PROCESSED
6. User received model score, and LLM explanation

Figure 3: Inference Pipeline Diagram

- Output: The processed entries are saved as a unified .jsonl file, where each record follows a schema containing a unique object identifier and the formatted contextual text, as seen in the below sample. This file is then uploaded to an S3 bucket and ingested into the Amazon Bedrock Knowledge Base for embedding generation.

  Each record follows the schema: { "id": "Object ID", "text": "Formatted contextual description" }

Wikidata Data Pipeline: This pipeline will run once to populate our S3 bucket with the relevant data.

- Ingestion: The pipeline programmatically queries the Wikidata SPARQL endpoint using artist-specific QIDs. Only artists included in the training and test sets are targeted, namely Vincent van Gogh, Johannes Vermeer, Frans Hals, Gerard ter Borch, Otto Wacker, and Han van Meegeren.
- Cleaning: Relevant biographical and stylistic attributes are extracted from the raw Wikidata results.
- Transformation: Each artist record is converted into a formatted natural language document optimized for embedding.
- Output: The processed artist entries are exported into a .jsonl file following the same schema as the MET data, with each record containing a unique Wikidata QID and the formatted contextual description, as seen in the record below. This approach ensures that both the datasets share a unified document structure within the RAG knowledge base, allowing for consistent embedding and retrieval

  Each record follows the schema: { "id": "Wikidata QID", "text": "Formatted artist contextual description" }

3.3 Inference Pipeline

Data Schemas: Refer to `src/apps/data_pipeline/schemas.py`

Pipeline Diagram:

Pipeline Description:

- Ingestion: The user uploads an image via our app. For reproducibility purposes, the image is also stored in S3 bucket for storing raw images during inference, and populated in DynamoDB with its metadata.
- Cleaning: The image is converted to RGB, and encoded as JPEG. Additionally, the user receives an error if the image is too small.
- Transformation: The image is divided into 2^p by 2^p equally sized patches, with p depending on the resolution of the original image as follows: p = 2, if the smaller side of an image is larger than 1024 pixels, and p = 1, if the smaller side is larger than 512 pixels and smaller than 1024. We also include a center-cropped square from the full image. All of these resulting sub-images are normalized to 256×256 pixels. This patching method allows the models to extract very fine-grained brushstroke level information from the smaller patches, but also more compositional and representational features from the larger patches.
- Data lake/warehouse design: For reproducibility purposes, each patch is stored in S3 bucket for storing processed images during inference, and its metadata is uploaded to DynamoDB.
- Using the patches stored in memory, and Swin transformer's model weights from Modal volume, we make an inference. We pass the feature map, and inference score to Claude Sonnet 4.5 LLM, which uses RAG to provide both the inference score, and a valid justification to the user.

Next Steps:

- Some patches may contain some superficial artifacts, due to the digital JPEG encoding. As this problem might hinder the training, we can apply a blurring filter (i.e., Gaussian blur) to the patches during data processing. This way, it creates an image that is less hazy than the original one.
- Furthermore, some of the patches may contain very minimal information (i.e. patches containing background), which could yield arbitrary predictions. We could use an entropy measure to quantify the amount of information contained in each patch, calculate the entropy of each individual color channel (RGB) separately for each patch and take the average of the three entropies, and filter out uninformative patches.
- Integrate the Smithsonian Open Access Database: Connect the S3 bucket containg the data from the Smithsonian museum to the RAG data preprocessing pipeline so that the data can be cleaned and formatted to ensure consistency with existing MET and Wikidata documents.
- Finalize Bedrock Embedding Model Selection: Decide which Amazon Bedrock model will be used to generate embeddings for our knowledge base

4 Part Four: Infrastructure as Code Implementation

Refer to infrastructure folder.

5 Part Five: Disaster Recovery Demonstration

Refer to infrastructure ReadMe.

Sources

[1]. https://zenodo.org/records/10276928

[2]. https://data.rijksmuseum.nl/

[3]. https://commons.wikimedia.org/wiki/Main$_{Page}$

[4]. https://github.com/metmuseum/openaccess

[5]. https://registry.opendata.aws/smithsonian-open-access/