EgyptianGen: A Text-to-Image Generation System for Ancient Egyptian Iconography and Scenes

Rabbia Waheed[1], Ummamma Nisar[2], Syed Furqan Ali[3]
[1,2,3]School of Mechanical and Manufacturing Engineering,
National University of Sciences and Technology (NUST),
Islamabad Pakistan

Abstract: This paper introduces EgyptianGen, a novel text-to-image generation system specifically designed to produce historically and culturally informed imagery of Ancient Egypt. The system leverages a multi-stage pipeline that includes: (1) processing of academic PDF sources for text and image extraction to build a domain-specific knowledge base; (2) creation and integration of a comprehensive Egyptian Academic Cultural Database detailing deities, pharaohs, artistic conventions, and more; (3) an open-source figure classification model (using SentenceTransformer) to identify Egyptian archetypes from user prompts; (4) an open-source prompt enhancement model (Flan-T5) that enriches user descriptions with contextual details from the cultural database; and (5) the FLUX text-to-image generation models (via Replicate API) to produce the final artwork. The system is accessible through a Gradio-based user interface, providing users with generated images, enhanced prompts, relevant academic context, and reference images. EgyptianGen aims to provide a valuable tool for education, research, and artistic exploration related to Ancient Egyptian heritage by bridging the gap between advanced AI capabilities and deep cultural knowledge.

***Index Terms*—Ancient Egypt, Text-to-Image Generation, AI in Cultural Heritage, Egyptian Mythology, Prompt Engineering**

I. Introduction

The field of artificial intelligence has witnessed remarkable advancements in generative models, particularly in text-to-image synthesis. Models based on Generative Adversarial Networks (GANs) and, more recently, diffusion models have demonstrated an extraordinary ability to generate coherent, high-resolution images from textual descriptions. These technologies are rapidly transitioning from research novelties to practical tools with diverse applications. Concurrently, there is a burgeoning interest in applying AI to creative industries and cultural heritage, offering new ways to engage with, understand, and preserve historical and artistic legacies. Text-to-image generation, in this context, holds the potential to visually reconstruct historical scenes, artifacts, and iconographies, making them more accessible and engaging for a wider audience. Ancient Egyptian art and iconography represent a rich tapestry of human history, offering profound insights into the civilization's beliefs, rituals, and daily life. This visual heritage is crucial for cultural preservation, education, and ongoing scholarly research. However, generating accurate, culturally sensitive, and historically informed depictions of Ancient Egypt using general-purpose text-to-image models presents significant challenges. These models, while powerful, often lack the nuanced, domain-specific knowledge required to correctly interpret prompts related to specific historical figures (e.g., deities, pharaohs with their unique attributes), artistic conventions (e.g., the canon of proportion, symbolic colors), and intricate iconographic details prevalent in Egyptian art. Generic models may produce anachronistic, stereotypical, or misleading imagery, thereby diminishing their utility for serious cultural applications. There is, therefore, a distinct need for a specialized tool that integrates deep, curated domain knowledge into the image generation pipeline to address these shortcomings. The primary goal of this project is to develop "EgyptianGen," a system capable of generating high-quality images from text prompts, with a specific focus on Ancient Egyptian themes, figures, symbols, and artistic styles. The system aims to move beyond generic image synthesis by emphasizing the integration of academic and cultural knowledge directly into the generation process. This involves creating a robust knowledge base derived from scholarly sources and leveraging it to guide AI models in interpreting user inputs and rendering images that are not only visually compelling but also historically and culturally resonant. EgyptianGen seeks to improve the relevance, accuracy, and educational value of AI-generated imagery in the context of Ancient Egyptian heritage.

A. System Overview

EgyptianGen employs a modular, multi-stage architecture to achieve objectives. The key components include:

- **PDF Processing:** An automated pipeline for extracting text and relevant images from scanned academic PDF books on Ancient Egypt, forming the raw material for the cultural database.
- **Egyptian Academic Cultural Database:** A structured JSON database containing curated information on historical sources, deity and pharaonic archetypes, commoner figures, artistic conventions, architectural elements, funerary practices, technical art quality markers, and the digitized content from processed PDFs.
- **Figure Classification:** An open-source Sentence Transformer-based model (all-MiniLM-L6-v2) fine-tuned or utilized for classifying user text prompts into predefined Ancient Egyptian figure archetypes (e.g., specific deities, types of pharaohs) [4].
- **Prompt Enhancement:** An open-source language model (google/flan-t5-base) that enriches the user's initial prompt with detailed historical, cultural, and artistic con-

text drawn from the cultural database and classified figure type.

- **Image Generation:** Utilization of state-of-the-art FLUX text-to-image models (flux-dev and flux-schnell) accessed via the Replicate API, which receive the enhanced, context-rich prompts [9].
- **User Interface:** A web-based interface built with Gradio[11], allowing users to input descriptions, select generation parameters, view the generated image, and access the enhanced prompt, associated academic context, and relevant reference images from the database.

This system makes use of both locally managed open-source models and external proprietary APIs for cutting-edge image synthesis.

II. Methodology

This section discusses the methodology of the project step by step.The block diagram outlining the major steps is shown in the figure 1 below:

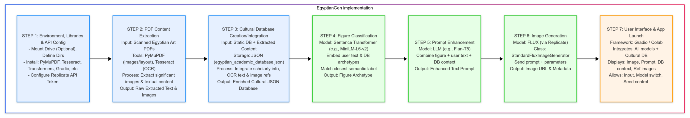

EgyptianGen implementation
STEP 1: Environment, Libraries & API Config - Mount Drive (Optional), Define Dirs - Install: PyMuPDF, Tesseract, Transformers, Gradio, etc. - Configure Replicate API Token
STEP 2: PDF Content Extraction Input: Scanned Egyptian Art PDFs Tools: PyMuPDF (images/layout), Tesseract (OCR) Process: Extract significant images & textual content Output: Raw Extracted Text & Images
STEP 3: Cultural Database Creation/Integration Input: Static DB + Extracted Content Storage: JSON (egyptian_academic_database.json) Process: Integrate scholarly info, OCR text & image refs Output: Enriched Cultural JSON Database
STEP 4: Figure Classification Model: Sentence Transformer (e.g., MiniLM-L6-v2) Embed user text & DB archetypes Match closest semantic label Output: Figure Archetype
STEP 5: Prompt Enhancement Model: LLM (e.g., Flan-T5) Combine figure + user text + DB context Output: Enhanced Text Prompt
STEP 6: Image Generation Model: FLUX (via Replicate) Class: StandardFluxImageGenerator Send prompt + parameters Output: Image URL & Metadata
STEP 7: User Interface & App Launch Framework: Gradio / Colab Integrates: All models + Cultural DB Displays: Image, Prompt, DB context, Ref images Allows: Input, Model switch, Seed control

Fig. 1. Block Diagram of EgyptianGen Implementation

A. Environment Setup and Configuration

The development and operational environment for EgyptianGen is designed for flexibility, supporting both local workspaces and cloud-based platforms like Google Colab. A standardized directory structure (EgyptianGen Workspace) is established to manage project files, including input data (PDFs), output data (extracted text, images, generated artwork), model caches, and configuration files. Paths are dynamically configured to ensure portability. Efficient package management is maintained through the configuration of a pip cache, which speeds up the installation of dependencies by reusing previously downloaded packages, crucial for iterative development and deployment in environments with ephemeral storage.

B. Package Installation and Library Imports

EgyptianGen relies on a comprehensive suite of Python libraries and packages. Standard Python libraries such as os, glob, json, requests, re, pickle, datetime, and traceback are used for file system operations, data handling, network requests, regular expressions, object serialization, and error management. For Google Colab integration, google.colab utilities are employed (e.g., for Google Drive mounting, if opted by the user). The user interface is built using gradio. The replicate library facilitates interaction with the FLUX image generation API. Image manipulation and processing are handled by Pillow (PIL). Core machine learning functionalities are provided by transformers and torch (for Hugging Face models like Flan-T5), sentence-transformers (for the figure classification model), and scikit-learn (for utilities like cosine similarity). PDF processing and Optical Character Recognition (OCR) capabilities are enabled by PyMuPDF (fitz) and pytesseract, respectively. Although not central to the final image generation logic presented, rembg and onnxruntime are included for potential background removal tasks. Numerical operations extensively use numpy. Interactive elements within Colab notebooks are supported by ipywidgets and IPython.display.

C. API Configuration

Access to the powerful FLUX text-to-image generation models (flux-dev and flux-schnell) is managed via the Replicate platform, requiring a REPLICATE API TOKEN. The system is designed to securely read this token, typically from a local configuration file (secrets.json) or, if not found, by prompting the user for manual input. This ensures that sensitive API credentials are not hardcoded and can be managed appropriately by the user.

D. EgyptianGen PDF Processor: Text and Image Extraction

A critical preparatory step in building EgyptianGen's knowledge base is the automated processing of academic literature.

- **Objective:** To systematically extract textual content and significant visual elements (images, diagrams) from scanned PDF books and articles focused on Ancient Egypt.
- **Input:** PDF files are placed by the user into a designated PDF INPUT DIR within the project workspace.
- **Process:** The system iterates through each PDF document page by page using the fitz library (PyMuPDF). For each page:

The page is rendered as a high-resolution image (get pixmap) and temporarily stored using Pillow. pytesseract is then applied to this image to perform OCR, converting the visual text into machine-readable strings. The fitz library is also used to directly extract embedded images from the PDF page object.The extracted textual content from each PDF is aggregated and saved into corresponding .txt files within a TEXT OUTPUT DIR, organized by the source PDF's name.

Extracted images are saved into subdirectories within an IMAGES OUTPUT DIR, again named after the source PDF, allowing for easy traceability. To prevent redundant processing of large PDF files, which can be time-consuming, a marker file (e.g., PDF Processed Log textfile, referred to as PROCESSED MARKER FILENAME in the concept) is maintained. This log tracks successfully processed PDFs, ensuring that they are skipped in subsequent runs unless explicitly overridden.

E. Egyptian Academic Cultural Database

The heart of EgyptianGen's ability to generate culturally informed imagery is its specialized knowledge base. To create and maintain a structured, machine-readable database (egyptian academic database json) that encapsulates key information about Ancient Egyptian culture, art, and history. The database is json in the project's data directory. The JSON database is organized into several key sections:

- **Historical sources:** Lists primary references, important ancient textual sources (e.g., Pyramid Texts, Coffin Texts, Book of the Dead), and significant museum collections or archaeological sites.
- **Deity archetypes:** Provides detailed entries for major and minor deities (e.g., Ra-Horakhty, Isis, Anubis, Thoth). Each entry includes fields for epithets, domains of influence, historical descriptions, detailed iconography (typical form, headdress, attire, sacred animals, symbols, common colors), and major cult centers. pharaonic archetypes: Describes different types or representations of pharaohs (e.g., Old Kingdom Founder, New Kingdom Warrior Pharaoh, Female Pharaoh). Entries include historical context, typical regalia (crowns like Nemes, Khepresh; scepters; false beard), and symbolic postures.
- **Commoner archetypes:** Details for other societal roles (e.g., Scribe, Artisan, Farmer, Priest), including their typical depictions, tools, and attire.
- **Artistic conventions:** Describes core principles of Egyptian art, such as perspective (frontalism, profile views), scale hierarchy, use of registers, and color symbolism.
- **Architectural elements:** Information on characteristic architectural forms like temples (pylons, hypostyle halls), tombs (mastabas, pyramids, rock-cut tombs), and obelisks.
- **Funerary practices:** Details on mummification, canopic jars, sarcophagi, and important funerary rituals or beliefs.
- **Technical specifications:** Includes a list of artistic quality markers – phrases designed to guide the image generation model towards desired aesthetic outcomes (e.g., ”museum-quality ancient Egyptian art,” ”stylized Egyptian canon of proportion,” ”detailed hieroglyphic inscriptions,” ”painted limestone relief”).
- **Digitized book contents:** This section is dynamically populated. It stores the textual content extracted from academic PDFs by the EgyptianGen PDF Processor, organized by book title, providing a rich corpus for contextual information retrieval.

The system loads this database at runtime. If the JSON file doesn't exist, a template might be created. The core static definitions (archetypes, conventions) are typically manually curated or seeded, while the digitized book contents section is updated programmatically by the PDF processing pipeline.

F. Text Snippet Retrieval Block

To leverage the textual data extracted from academic sources, a dedicated helper function is implemented. This function is designed to search within the digitized book contents section of the cultural database (and potentially other textual fields of archetypes) to find and return text snippets that are most relevant to a given set of search terms (often derived from a user's prompt or a classified figure). It provides a list of keywords or phrases to search for. ,the dictionary of texts extracted from PDFs, The full cultural database, for broader context, limits the number of snippets extracted from any single book source, defines the approximate length (e.g., in characters or words) of each extracted snippet, a boolean flag, potentially to prioritize snippets from sources marked as primary or more authoritative. The retrieved snippets provide rich, context-specific textual information that is fed into both the figure classification stage (to build more descriptive embeddings for archetypes) and the prompt enhancement stage (to provide the LLM with relevant details for enriching the final image generation prompt). This mechanism directly connects the image generation process to the knowledge contained in the processed academic literature.

G. Open Source Figure Classification Model

A key innovation in EgyptianGen is its ability to understand the specific Ancient Egyptian figures or concepts mentioned in a user's prompt.The objective is to accurately classify a user's natural language input into one of the predefined Ancient Egyptian figure archetypes (deities, pharaohs, commoners) stored in the EGYPTIA ACADEMIC DB. The classification leverages a pre-trained SentenceTransformer model, specifically all-MiniLM-L6-v2, known for its efficiency and effectiveness in generating sentence embeddings for semantic similarity tasks.The SentenceTransformer model is loaded. To optimize loading times, a local caching mechanism (ModelsCache) is implemented, storing downloaded models for reuse across sessions. For each archetype defined in the EGYPTIAN ACADEMIC DB (e.g., ”Isis,” ”New Kingdom Warrior Pharaoh”), a comprehensive descriptive text is dynamically constructed. This text combines the structured information from the archetype's database entry (like historical descriptions, epithets, domains, iconography, symbols, regalia) with relevant contextual snippets retrieved (using the archetype's name or key terms as search queries).

These rich descriptions are then encoded by the SentenceTransformer model to produce a dense vector embedding for each archetype.

The resulting set of (archetype label, archetype embedding) pairs is stored as figure embeddings data and cached to a .pkl file to avoid re-computation on subsequent runs. When a user provides an input text, it is first encoded using the same loaded SentenceTransformer model to get its embedding. The cosine similarity is then calculated between this user input embedding and all the pre-computed archetype embeddings stored in figure embeddings data. The archetype corresponding to the embedding with the highest cosine similarity score is identified as the most likely classification. A confidence threshold 0.25 is applied to this similarity score. If the score exceeds the threshold, the classification is considered successful. If the highest semantic similarity score is below the confidence threshold, or if the embedding model fails for some reason, a fallback mechanism is triggered. This fallback relies on keyword-based matching. It iterates through a predefined keyword mapping that links specific keywords (e.g., 'ra', 'sun god', 'isis', 'pharaoh', 'scribe') directly to their corresponding archetypes. The user's input text is scanned for these keywords.

This ensures that even if semantic understanding is weak, common and explicit references to figures can still be caught. The system dynamically uses the list of valid archetypes currently present in the loaded cultural database for this mapping. The classifier returns the identified figure type (e.g., ”Deity Isis,” ”Pharaoh NewKingdomWarrior”) and the method of classification (semantic or keyword).

H. Open Source Prompt Enhancement Model

Once a figure or theme is identified, the initial user prompt is enriched to guide the image generation model more effectively.To transform a potentially simple user prompt into a detailed, historically and culturally nuanced description suitable for generating high-quality, contextually appropriate images of Ancient Egypt. The primary enhancement mechanism uses google/flan t5 base, a versatile text-to-text generation model from Hugging Face, known for its instruction-following capabilities. This model is also loaded via the ModelsCache. The method first receives the figure type as determined by the EgyptianFigureClassifier. Detailed information (e.g., historical overview, iconographic details like attire, symbols, typical colors, notes on artistic representation, major cult centers) is retrieved directly from the corresponding entry in the EGYPTIAN ACADEMIC DB. Additional relevant textual context is fetched from the digitized book contents using the ”get relevant text snippets” function, using the figure’s name and related terms as queries.A carefully constructed ”meta prompt” or context prompt is assembled to instruct the Flan-T5 model. This meta-prompt typically includes a clear task definition (e.g., ”Transform the user’s request into a detailed, vivid, and historically inspired image generation prompt focusing on Ancient Egyptian art and culture.”) and specific instructions to the LLM, guiding it to emphasize visual details like posture, attire, symbolic objects, characteristic artistic conventions (e.g., profile views, hieroglyphic elements), and relevant historical period styles.

This comprehensive meta prompt is then fed to the Flan-T5 pipeline.

The output from Flan-T5, which is the LLM-generated enhanced prompt, undergoes a simple post-processing step to ensure that the user’s original core request is preserved and integrated naturally within the enhanced text. This fallback mechanism is used if the LLM-based enhancement is disabled or if the LLM fails. Regardless of whether LLM or rule-based enhancement was used, this final step appends crucial artistic and quality directives to the prompt.

It selects relevant phrases from the artistic quality markers list in the technical specifications section of the cultural database (e.g., ”museum-quality ancient Egyptian art,” ”stylized Egyptian canon of proportion,” ”vibrant mineral pigments,” ”detailed hieroglyphic inscriptions”).

General high-quality descriptive terms (e.g., ”highly detailed masterpiece,” ”sharp focus,” ”cinematic lighting”) are also added to encourage the generation of visually impressive images. The method returns the fully enhanced text prompt, ready to be sent to the image generation model, along with the academic context used for transparency.

I. FLUX Generation System

This component is responsible for the actual synthesis of images from the carefully crafted prompts.It generates high quality images corresponding to the enhanced text prompts using the advanced FLUX text-to-image models. The ”black forest labs/flux dev” is referred to as the higher quality, more detailed model, though potentially slower and more computationally intensive (higher cost on Replicate).The ”black forest labs/flux schnell” is presented as a faster, ”preview” quality model, suitable for quicker iterations. The user, via the UI, selects either the dev or schnell FLUX model variant. This choice determines the specific Replicate API endpoint to be called. The system prepares the input payload for the Replicate API. Key parameters include:

- **Prompt:** The fully enhanced text prompt generated by the EgyptianPromptEnhancer.
- **Aspect ratio:** User-selectable, e.g., ”1:1”, ”16:9”, ”9:16”. Defaults to ”1:1” if not specified.
- **Output format:** Typically ”png” or ”jpeg”. Defaults to ”png”.

For the flux-dev model, additional control parameters might be exposed or set, such as:

- **Guidance scale:** Controls how strictly the model adheres to the prompt.
- **Inference steps:** Affects image quality and generation time.
- **Seed:** An optional seed value can be provided by the user. Using the same seed with the same prompt and parameters aims for reproducible image outputs. If no seed is provided, a random one is typically used.

The replicate.run() function is invoked with the chosen model’s Replicate ID and the prepared input parameters. This is a blocking call that waits for the image generation to complete on Replicate’s servers. Upon successful generation, the Replicate API returns a URL (or a list of URLs if multiple images were requested, though EgyptianGen seems to focus on single image outputs per request) pointing to the generated image(s). The system then uses the requests library to download the image from this URL. The image data is opened using Pillow for potential further processing or saving. A generation history (likely a list of dictionaries or objects) is maintained. Each entry records details of the generation request: timestamp, FLUX model used, the full prompt sent, the returned image URL, the estimated cost (if provided by Replicate or estimated), the seed used, and the enhancement method employed. This is valuable for debugging, user history, and cost management.

J. User Interface (EgyptianGen)

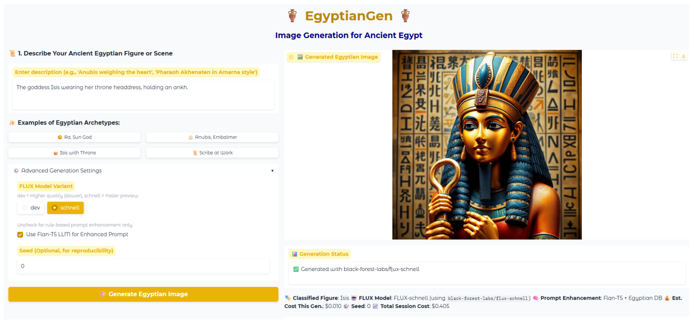
To make the EgyptianGen system accessible and interactive, user interfaces are provided for both web and Colab environments. Gradio is used to create an intuitive web-based application. The interface is typically structured with a prominent Title (”EgyptianGen”) and a descriptive header explaining the tool’s purpose, a Textbox for the user to type their textual description of the desired image, convenience Buttons with example archetypes (e.g., ”Isis,” ”Anubis,” ”Warrior Pharaoh”) to quickly populate the input textbox with common requests, an Accordion or collapsible section for ”Advanced Settings,” containing: Radio buttons or a Dropdown to select the FLUX model variant (dev or schnell), a Checkbox to toggle LLM-based prompt enhancement,and a Number input for specifying a seed for reproducible generation.

A main ”Generate Image” Button to trigger the image creation process and image component is added to display the generated artwork. A Textbox for generation status is provided= for real-time feedback (e.g., ”Classifying figure...”, ”Enhancing prompt...”, ”Generating image...”).

A Markdown component to display detailed ”Generation Information”: the classified figure, the FLUX model used, the enhancement method (LLM or rule-based), estimated cost, and the seed..

An Accordion for ”Academic Context and Enhanced Prompt,” displays the full, enhanced prompt that was sent to the FLUX model.The academic context (e.g., snippets from the cultural database, iconography details) used during prompt enhancement. This ”generate egyptian avatar via text” function is the main backend function orchestrated by Gradio. It takes user inputs from the UI, calls the various EgyptianGen modules in sequence (classification, enhancement, image generation), and returns all outputs (generated image, status messages, academic context, reference images, generation details) to be displayed in the Gradio interface. It supports Gradio’s progress object for iterative updates to the UI during long operations. For users working directly within a Google Colab notebook or a similar Jupyter environment, ipywidgets and IPython.display are used to create a simpler, embedded interactive interface. This typically involves creating widgets like Text, Dropdown, Checkbox, and Button that mirror the core input options of the Gradio UI (user description, model choice, LLM toggle, seed). When the ”Generate” button in this widget interface is clicked, it calls the same underlying ”generate egyptian avatar via text” function.

The generated image and textual outputs (enhanced prompt, context) are displayed directly within the output area of the Colab cell using IPython.display.Image and IPython.display.Markdown. This provides a convenient way to test and interact with the system without launching the full Gradio web server. A utility function is often included, especially for Gradio applications, to find and kill any existing processes that might be occupying the default port (e.g., 7860) before attempting to launch a new Gradio app. This helps avoid ”port already in use” errors.


EgyptianGen
Image Generation for Ancient Egypt
1. Describe Your Ancient Egyptian Figure or Scene
Enter description (e.g., 'Anubis weighing the heart', 'Pharaoh Akhenaten in Amarna style')
Examples of Egyptian Archetypes:
Ra, Sun God
Anubis, Embalmer
Isis with Throne
Scribe at Work
Advanced Generation Settings
FLUX Model Variant
dev = Higher quality (slower), schnell = Faster preview
Uncheck for rule-based prompt enhancement only.
Seed (Optional, for reproducibility)
Generate Egyptian Image
Generated Egyptian Image
Generation Status

The goddess Isis wearing her throne headdress, holding an ankh.

dev [✓ schnell]

[✓] Use Flan-T5 LLM for Enhanced Prompt

0

Generated with black-forest-labs/flux-schnell

Classified Figure: Isis **FLUX Model**: FLUX-schnell (using `black-forest-labs/flux-schnell`) **Prompt Enhancement**: Flan-T5 + Egyptian DB **Est. Cost This Gen.**: $0.010 **Seed**: 0 **Total Session Cost**: $0.405

Fig. 2. Block Diagram of EgyptianGen Implementation

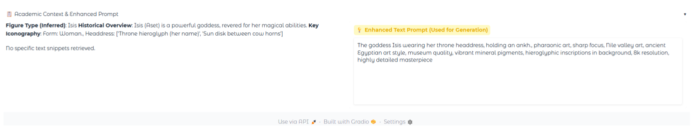

Academic Context & Enhanced Prompt
Enhanced Text Prompt (Used for Generation)
Use via API · Built with Gradio · Settings

Figure Type (Inferred): Isis **Historical Overview**: Isis (Aset) is a powerful goddess, revered for her magical abilities. **Key Iconography**: Form: Woman., Headdress: ['Throne hieroglyph (her name)', 'Sun disk between cow horns']

No specific text snippets retrieved.

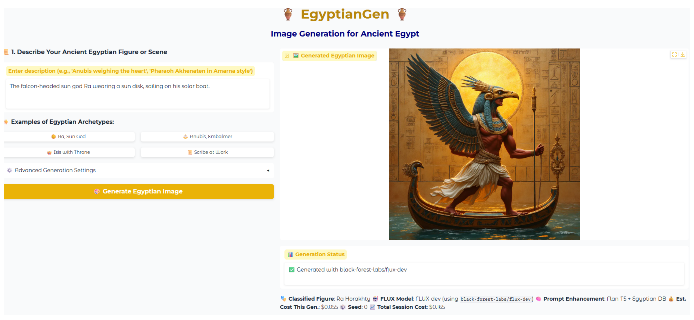
The goddess Isis wearing her throne headdress, holding an ankh., pharaonic art, sharp focus, Nile valley art, ancient Egyptian art style, museum quality, vibrant mineral pigments, hieroglyphic inscriptions in background, 8k resolution, highly detailed masterpiece

Fig. 3. Block Diagram of EgyptianGen Implementation

III. Results and Discussion

This section outlines the expected results and discussion points based on the system’s design. Actual execution and evaluation would be needed to provide concrete findings. It is anticipated that EgyptianGen would produce a diverse range of images showcasing its capabilities. Examples would be presented, demonstrating:

- **Deity Depictions:** Images of major deities shown in figure 4 like Ra-Horakhty (falcon-headed, sun disk), Isis (throne headdress or cow horns with sun disk, often winged), Anubis (jackal-headed, associated with mummification), and Thoth (ibis-headed or baboon form, scribe’s palette). These images would aim to reflect their known iconographic attributes as detailed in the EACD.


EgyptianGen
Image Generation for Ancient Egypt
1. Describe Your Ancient Egyptian Figure or Scene
Enter description (e.g., 'Anubis weighing the heart', 'Pharaoh Akhenaten in Amarna style')
Examples of Egyptian Archetypes:
Ra, Sun God
Anubis, Embalmer
Isis with Throne
Scribe at Work
Advanced Generation Settings
Generate Egyptian Image
Generated Egyptian Image
Generation Status

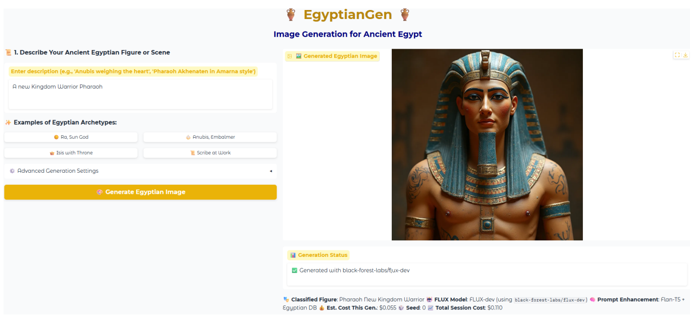
The falcon-headed sun god Ra wearing a sun disk, sailing on his solar boat.

Generated with black-forest-labs/flux-dev

Classified Figure: Ra Horakhty **FLUX Model**: FLUX-dev (using `black-forest-labs/flux-dev`) **Prompt Enhancement**: Flan-T5 + Egyptian DB **Est. Cost This Gen.**: $0.055 **Seed**: 0 **Total Session Cost**: $0.165

Fig. 4. Block Diagram of EgyptianGen Implementation

- **Pharaonic Representations:**Depictions of pharaohs shown in figure 5 in various contexts, such as a ”New Kingdom Warrior Pharaoh” in a chariot, or a pharaoh making offerings, showcasing appropriate regalia (e.g., Nemes headdress, Khepresh crown, crook and flail).


EgyptianGen
Image Generation for Ancient Egypt
1. Describe Your Ancient Egyptian Figure or Scene
Enter description (e.g., 'Anubis weighing the heart', 'Pharaoh Akhenaten in Amarna style')
Examples of Egyptian Archetypes:
Ra, Sun God
Anubis, Embalmer
Isis with Throne
Scribe at Work
Advanced Generation Settings
Generate Egyptian Image
Generated Egyptian Image
Generation Status

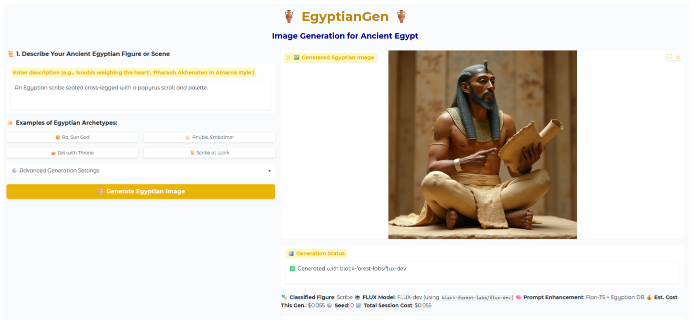
A new Kingdom Warrior Pharaoh

Generated with black-forest-labs/flux-dev

Classified Figure: Pharaoh New Kingdom Warrior **FLUX Model**: FLUX-dev (using `black-forest-labs/flux-dev`) **Prompt Enhancement**: Flan-T5 + Egyptian DB **Est. Cost This Gen.**: $0.055 **Seed**: 0 **Total Session Cost**: $0.110

Fig. 5. Block Diagram of EgyptianGen Implementation

- **Scenes and Themes:** Generated scenes shown in figure 6 like ”a scribe writing on papyrus in a temple scriptorium,” ”a funerary procession with canopic jars,” or ”an offering scene before an altar,” incorporating elements from artistic conventions and architectural elements of the EACD.


EgyptianGen
Image Generation for Ancient Egypt
1. Describe Your Ancient Egyptian Figure or Scene
Examples of Egyptian Archetypes:
Ra, Sun God
Anubis, Embalmer
Isis with Throne
Scribe at Work
Advanced Generation Settings
Generate Egyptian Image
Generated Egyptian Image
Generation Status

Enter description (e.g., 'Anubis weighing the heart', 'Pharaoh Akhenaten in Amarna style')

An Egyptian scribe seated cross-legged with a papyrus scroll and palette.

Generated with black-forest-labs/flux-dev

Classified Figure: Scribe **FLUX Model**: FLUX-dev (using `black-forest-labs/flux-dev`) **Prompt Enhancement**: Flan-T5 + Egyptian DB **Est. Cost This Gen.**: $0.055 **Seed**: 0 **Total Session Cost**: $0.055

Fig. 6. Block Diagram of EgyptianGen Implementation

A comparative showcase of images generated by FLUX-dev (expected to be higher detail and fidelity) versus FLUX-schnell (expected to be faster, potentially less refined) for the same enhanced prompts would illustrate the trade-offs.

6.3. Performance Expected generation times for typical prompts using FLUX-dev versus FLUX-schnell via the Replicate API shows that FLUX-schnell is 30/40 seconds faster. This would also acknowledge that these times are subject to API load and network latency. Time taken for local processes like figure classification and prompt enhancement (especially LLM-based) is very less making it efficient for use, along with the benefits of caching models and pre-computed embeddings for improved responsiveness on subsequent runs. While a one-time or infrequent task, the efficiency of the PDF processing module would be noted, particularly its ability to handle multiple documents and avoid reprocessing.

and Limitations A section will discuss the system's challenges and limitations:

- **Database Dependency:** The quality and comprehensiveness of the Egyptian Academic Cultural Database are paramount. Gaps, inaccuracies, or biases in the curated data or the processed PDF content will directly impact the system's output. Maintaining and expanding this database is a significant ongoing effort.
- **Model Biases and Inaccuracies:** Pre-trained AI models (SentenceTransformer, Flan-T5, FLUX) may carry inherent biases from their training data, which could manifest in the generated images or text. They might also hallucinate details or struggle with highly specific or obscure iconographic elements not well-represented in their training.
- **Nuance and Specificity:** While EgyptianGen aims for cultural accuracy, achieving perfect representation of all nuances of Ancient Egyptian art (which varied over millennia and regions) is extremely challenging. The FLUX models, though advanced, might still generalize or miss subtle artistic details.
- **OCR Accuracy:** The quality of text extracted via OCR from scanned PDFs can vary depending on the PDF's scan quality, layout complexity, and font. Errors in OCR can propagate into the cultural database and affect downstream tasks.
- **Scope of Knowledge:** The system's knowledge is limited to the content of its database. It cannot reason about or depict aspects of Ancient Egypt not covered therein.

IV. Future Work

The current EgyptianGen system provides a strong foundation, but numerous avenues for future development and refinement exist. It involves:

- Continuously augment the egyptian academic database json with more archetypes (e.g., specific monsters, regional variations of deities), symbols, detailed architectural elements, explicit chronological markers, and a wider range of digitized texts.
- Incorporate more granular metadata for extracted images in the database, perhaps linking them directly to specific textual descriptions or archetypes.
- Explore fine-tuning the SentenceTransformer model on a domain-specific corpus related to Egyptology for even more accurate figure classification.
- Fine-tune the Flan-T5 model (or a larger equivalent) on a dataset of user prompts and expertly crafted enhanced prompts relevant to Ancient Egypt to improve its nuance and contextual understanding.
- Integrate more sophisticated image analysis techniques (e.g., object detection, style classification, feature extraction) for the images extracted from PDFs. This could help automatically tag images, identify common symbols, and further enrich the cultural database.
- Implement a system within the UI for users to rate the quality and accuracy of generated images and the relevance of the academic context. This feedback could be invaluable for identifying areas for improvement in the database and models.
- Extend the concept beyond 2D images to generate 3D models of artifacts, statues, or even reconstruct architectural scenes of Ancient Egyptian environments based on textual descriptions and database information.
- Adapt the system to handle prompts and display information in multiple languages, broadening its accessibility.
- Allow users more granular control over the artistic style, specific elements to include/exclude, or even to upload their own reference images to guide the generation process (image-to-image or image-prompted generation).

Hence,improving this software over time can make it more beneficial. .

V. Potential Applications

EgyptianGen, and systems like it, have a wide range of potential applications:

- **Education:** An interactive tool for students and educators at all levels to visualize and learn about Ancient Egyptian deities, pharaohs, daily life, art, and mythology in a more engaging way than static textbook images.
- **Research:** An aid for Egyptologists, art historians, and archaeologists to rapidly prototype visualizations of iconographic variations, reconstruct fragmented scenes, or explore "what-if" scenarios based on textual descriptions from ancient sources or scholarly hypotheses.

- **Artistic Inspiration and Creation:** A resource for artists, illustrators, and designers seeking historically-informed inspiration for creative projects, game development, or film production.
- **Museums and Cultural Heritage Institutions:** Potential for creating dynamic illustrative content for exhibitions, interactive displays, or online educational platforms, making collections more accessible and engaging.
- **Content Creation:** A tool for generating unique visual content for documentaries, publications, or online media related to Ancient Egypt.

VI. Conclusion

In conclusion, EgyptianGen introduces a specialized text-to-image generation system for Ancient Egypt, distinguished by its novel integration of a domain-specific cultural database to meticulously guide AI models through a multi-stage pipeline. This approach, encompassing context-aware figure classification and detailed prompt enhancement, significantly improves the contextual relevance and historical accuracy of generated imagery. By providing an end-to-end solution with an accessible user interface that also delivers academic context, EgyptianGen demonstrates a valuable method for bridging advanced AI capabilities with the nuanced requirements of cultural heritage, offering a potent tool for education, research, and artistic exploration within the rich domain of Ancient Egyptian studies.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022.

[2] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[3] H. W. Chung, S. Narang, O. Vinyals *et al.*, "Scaling instruction-finetuned language models," *arXiv preprint arXiv:2210.11416*, 2022.

[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[5] E. A. W. Budge, *The Gods of the Egyptians: Or, Studies in Egyptian Mythology*. London, UK: Methuen & Co., 1904.

[6] G. Hart, *The Routledge Dictionary of Egyptian Gods and Goddesses*. London, UK: Routledge, 2005.

[7] G. Robins, *The Art of Ancient Egypt*. Cambridge, MA: Harvard University Press, 2008.

[8] R. H. Wilkinson, *The Complete Gods and Goddesses of Ancient Egypt*. London, UK: Thames & Hudson, 2003.

[9] C. Shin, J. Choi, H. Kim, and S. Yoon, "Large-scale text-to-image model with inpainting is a zero-shot subject-driven image generator," Data Science and AI Laboratory, ECE, Seoul National Univ., 2023.

[10] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

[11] Gradio documentation. [Online]. Available: https://www.gradio.app

[12] PyMuPDF documentation. [Online]. Available: https://pymupdf.readthedocs.io

[13] Tesseract OCR. [Online]. Available: https://github.com/tesseract-ocr/tesseract

[14] Pillow documentation. [Online]. Available: https://pillow.readthedocs.io

[15] Hugging Face Transformers. [Online]. Available: https://huggingface.co/transformers