HORUS AI

Horus AI: Guardian of Ancient Egyptian Civilization

Smart exploration of ancient Egyptian artifacts unlocking silent stories.

History speaks again through AI's intelligent eyes.

Have You Ever Lived History?
History?

Egypt's Legacy

One of humanity's greatest civilizations

Beyond Sight

Experience history, not just see it

Technology's Role

Live history through advanced tech

Enter Artificial Intelligence

AI Today

From chatbots to self-driving cars

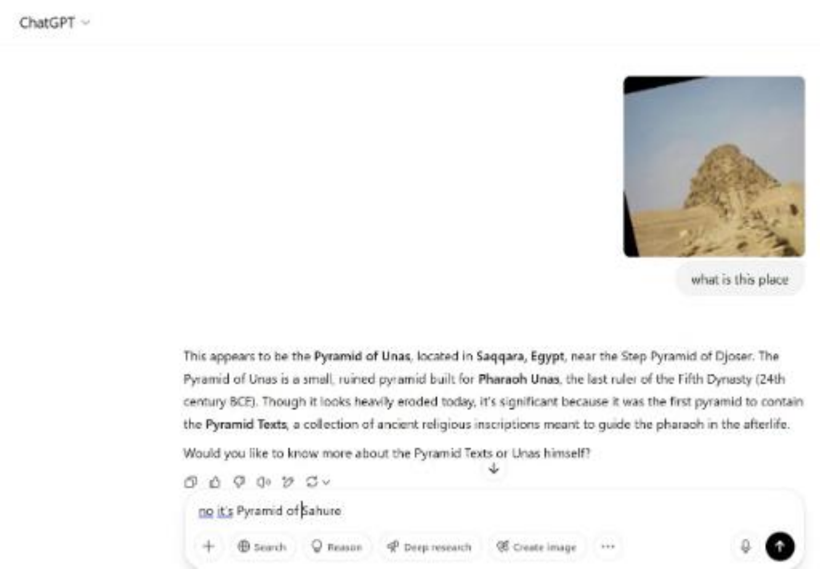
Past Connection

AI can't yet understand artifacts

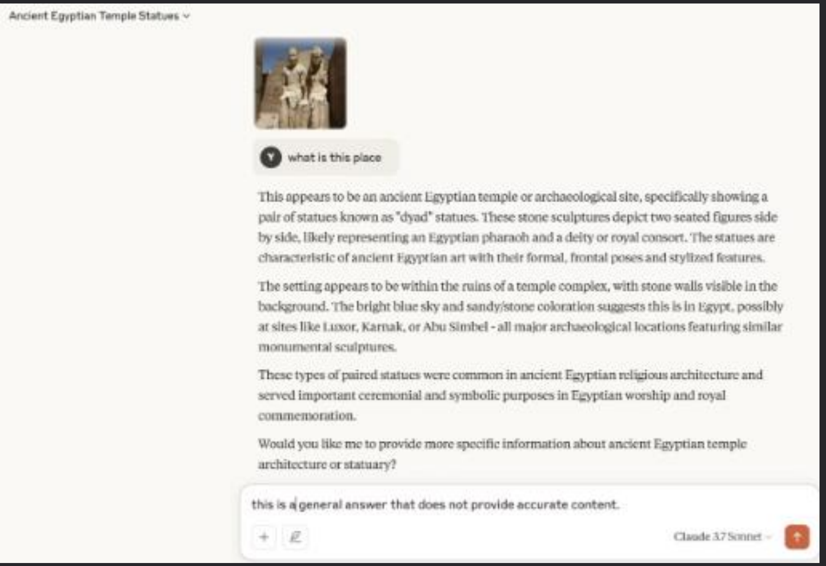
Horus AI

Teaching AI to understand history

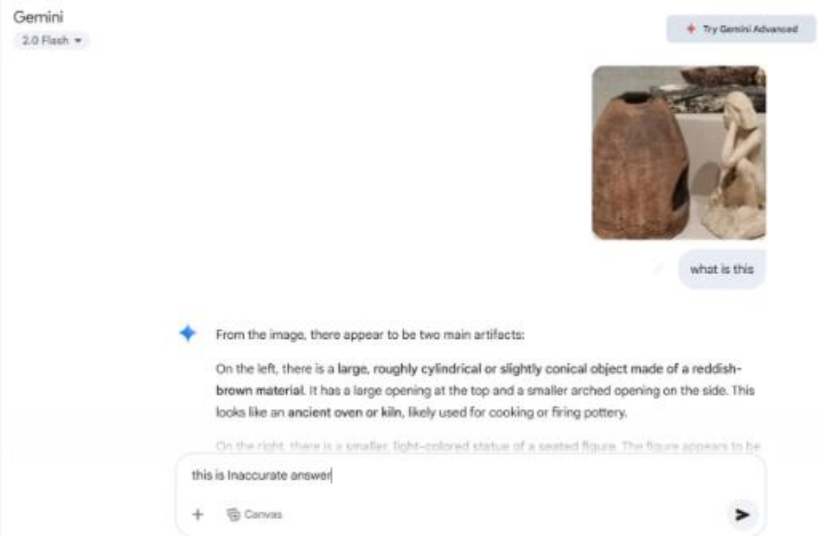
LLMs tend to provide general responses and sometimes sometimes misclassify images !



Chat GPT



Claude AI



Gemini

Horus Awakens

Image Classification

Accurate artifact identification

Historical Descriptions Descriptions

Informative and engaging content

Heritage Recommendations

Personalized site suggestions

Interactive Chat

Virtual archaeological guide

The Power Behind the Eye

Keras Model

Transfer learning for image classification

Google Gemini API API

Generates intelligent responses

Flask

User-friendly interactive web app

Modular Codebase

Efficient and maintainable structure

Your Journey With Horus

1. **Upload Image**

   User submits artifact photo

2. **Classification**

   AI identifies artifact

3. **Description**

   Generates historical context

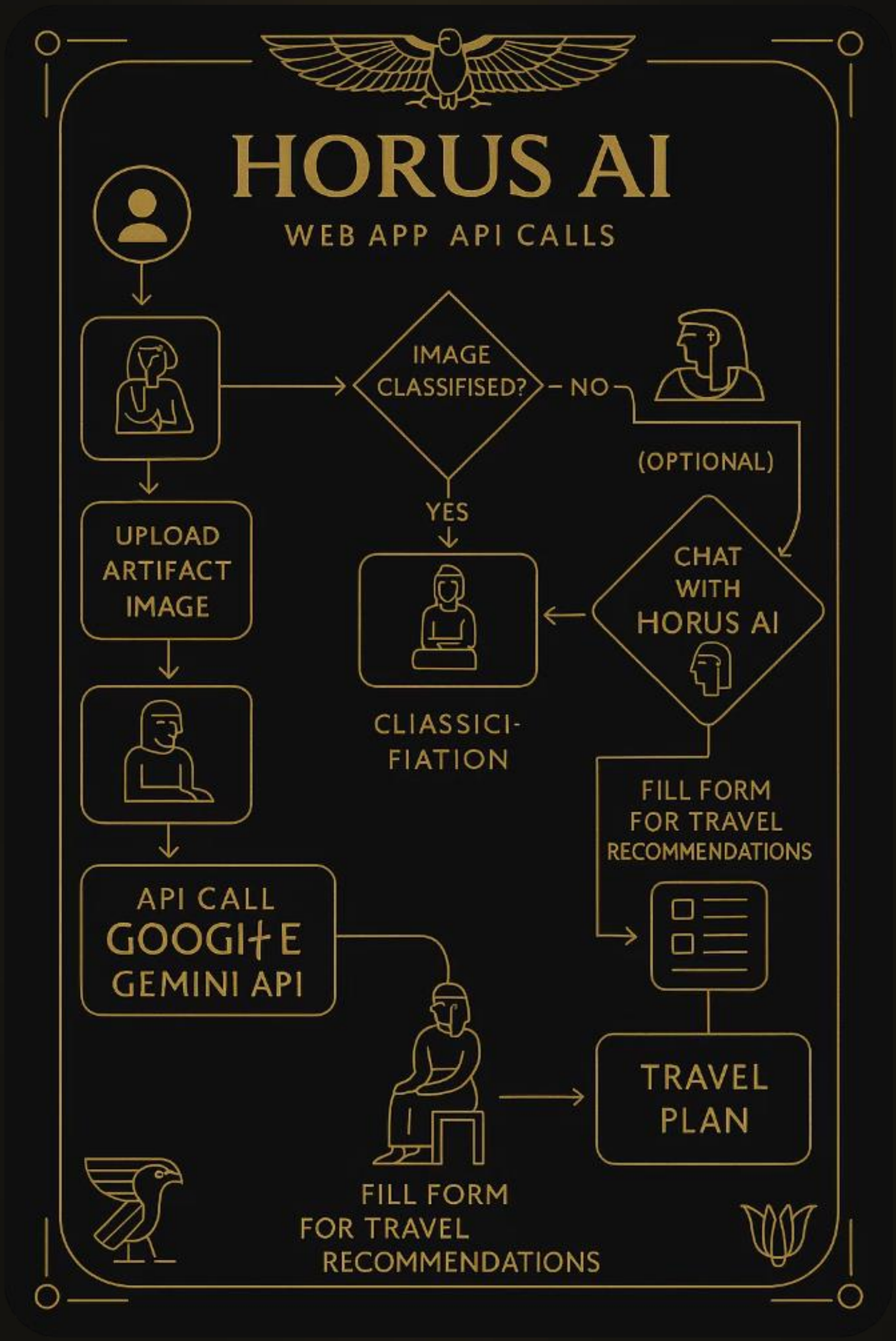
4. **Recommendations**

   Suggests travel destinations

5. **Live Q&A**

   Interactive AI guide chat

How The System Works?



Wisdom of the Eye

CNN Model

Transfer learning with preprocessing

Validation accuracy ~80%

Gemini API

Enhances explanations and interaction

Examples

- Anubis statue: Guardian of Tombs
- Philae Temple column: Roman heritage

Now let's Try The Model !

http://127.0.0.1:5000

Data Collection

We explored several artifact datasets and selected selected **The best three** based on their quality and and relevance to the project goals.

Sources

https://drive.google.com/drive/folders/1TYyceWyKaYWyKaYFqb0qI1LyjV3YJcbCjMw9M

Data Augmentation

Techniques

- Rotation and flipping
- Zooming and cropping
- Brightness and contrast shifts

Purpose

Enhance model robustness against limited data

Simulate real-world variations

Preprocessing & Augmentation Setup

This part defines the preprocessing pipeline using image transformations (rotation, flipping, color jitter, resizing), and resizes images to a uniform size while filtering out low-resolution samples — setting the foundation for effective learning

```
taset_path = "unzipped_data/cleaned_data"
rget_size = (224, 224)
n_resolution = (100, 100)
n_images = 100
x_images = 150
n_threshold = 10

gmentation = transforms.Compose([
  transforms.RandomHorizontalFlip(),
  transforms.RandomRotation(15),
  transforms.ColorJitter(brightness=0.2, contrast=0.2),
  transforms.RandomResizedCrop(target_size[0], scale=(0.8, 1.0))



f augment_image(image_path, save_path, index):
  image = Image.open(image_path).convert("RGB")
  augmented = augmentation(image)
  augmented = augmented.resize(target_size)
  augmented.save(f"{save_path}/aug_{index}.jpg")
r class_name in os.listdir(dataset_path):
  class_path = os.path.join(dataset_path, class_name)
  if not os.path.isdir(class_path):
      continue

  valid_images = []
  for img_name in os.listdir(class_path):
      img_path = os.path.join(class_path, img_name)
      try:
          img = Image.open(img_path).convert("RGB")
          w, h = img.size
          if w < min_resolution[0] or h < min_resolution[1]:
              os.remove(img_path)
              continue
          img = img.resize(target_size)
          img.save(img_path)
          valid_images.append(img_name)
      except:
          if os.path.exists(img_path):
              os.remove(img_path)
```

```python
                img.save(img_path)
                valid_images.append(img_name)
            except:
                if os.path.exists(img_path):
                    os.remove(img_path)

        image_count = len(valid_images)

        if image_count < min_threshold:
            shutil.rmtree(class_path)
            print(f"🗑 Removed class '{class_name}' (less than {min_threshold} images)")
            continue

        elif image_count > max_images:
            images_to_remove = image_count - max_images
            images_to_delete = random.sample(valid_images, images_to_remove)
            for img in images_to_delete:
                os.remove(os.path.join(class_path, img))
            print(f"📉 Reduced '{class_name}' to {max_images} images")

        elif image_count < min_images:
            images_needed = min_images - image_count
            valid_images = [img for img in os.listdir(class_path) if img.endswith(('.jpg', '.jpeg', '.png'))]
            if not valid_images:
                print(f"⚠ Skipping augmentation for '{class_name}' (no valid images found)")
                continue

            for i in range(images_needed):
                img_to_augment = random.choice(valid_images)
                augment_image(os.path.join(class_path, img_to_augment), class_path, i)
            print(f"🔄 Augmented '{class_name}' to {min_images} images")

print("\n✅ كل الكلاسات تم تنظيفها وتوحيد الصور وتوازن عدد الصور بينهم.")

new_data = {}
for class_name in os.listdir(dataset_path):
    class_path = os.path.join(dataset_path, class_name)
    if os.path.isdir(class_path):
        image_files = [os.path.join(class_path, img) for img in os.listdir(class_path) if img.endswith(('.jpg', '.jpeg', '.png'))]
        if image_files:  # التأكد من أن الكلاس يحتوي على صور
            new_data[class_name] = image_files
print("\n✅ تم حفظ الداتا الجديدة في المتغير new_data")
```

Data Cleaning and Balancing

This script filters images by resolution, removes underrepresented classes, limits classes with too many images, and performs data augmentation to balance the dataset — ensuring consistency and quality for model training.

Baseline Model

Architecture

Keras Convolutional Neural Network

Performance

Initial accuracy ~50%

Training

Standard setup with cross-entropy loss

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='softmax')
])

model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

model.fit(train_gen, epochs=10, validation_data=test_gen)

loss, acc = model.evaluate(test_gen)
print(f"\n✅ دقة النموذج على بيانات الاختبار: {acc * 100:.2f}%")
```

Model Code

Built and compiled a CNN model using TensorFlow and Keras with Conv2D, MaxPooling2D, Flatten, Dense, and Dropout layers. Used Adam optimizer and categorical crossentropy loss.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_4 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_5 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten_1 (Flatten)	(None, 86528)	0
dense_2 (Dense)	(None, 512)	44,302,848
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 206)	105,678

```
 Total params: 44,501,774 (169.76 MB)
 Trainable params: 44,501,774 (169.76 MB)
 Non-trainable params: 0 (0.00 B)
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset
  self._warn_if_super_not_called()
Epoch 1/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2397s 5s/step - accuracy: 0.0244 - loss: 5.1570 - val_accuracy: 0.1755 - val_loss: 4.1335
Epoch 2/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2370s 4s/step - accuracy: 0.1762 - loss: 3.9246 - val_accuracy: 0.2970 - val_loss: 3.3060
Epoch 3/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2370s 4s/step - accuracy: 0.3258 - loss: 2.9930 - val_accuracy: 0.3792 - val_loss: 2.7057
Epoch 4/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2385s 4s/step - accuracy: 0.4760 - loss: 2.1286 - val_accuracy: 0.4208 - val_loss: 2.5350
Epoch 5/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2403s 5s/step - accuracy: 0.6080 - loss: 1.5411 - val_accuracy: 0.4702 - val_loss: 2.3647
Epoch 6/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2383s 4s/step - accuracy: 0.6958 - loss: 1.1342 - val_accuracy: 0.4777 - val_loss: 2.3435
Epoch 7/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2386s 4s/step - accuracy: 0.7566 - loss: 0.8537 - val_accuracy: 0.4861 - val_loss: 2.3981
Epoch 8/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2377s 4s/step - accuracy: 0.8031 - loss: 0.6928 - val_accuracy: 0.4828 - val_loss: 2.4154
Epoch 9/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2377s 4s/step - accuracy: 0.8212 - loss: 0.6003 - val_accuracy: 0.4805 - val_loss: 2.6760
Epoch 10/10
532/532 ━━━━━━━━━━━━━━━━━━━━ 2399s 5s/step - accuracy: 0.8585 - loss: 0.4867 - val_accuracy: 0.4897 - val_loss: 2.5848
133/133 ━━━━━━━━━━━━━━━━━━━━ 154s 1s/step - accuracy: 0.4801 - loss: 2.6403

✅ دقة النموذج على بيانات الاختبار: 48.97%
```

Baseline model Result

Challenges & Accuracy Improvements

Challenges

- Overfitting on small dataset
- Class imbalance among artifact types

Solutions

- Hyperparameter tuning
- Enhanced data quality
- Architecture adjustments

Accuracy improved to 80%

Our journey to raise the accuracy from 50% to 80% (error analysis)

```
misclassified_df["True Label"].value_counts().head(20)
```

True Label	count
Elephantine	29
Statue of Ramesses II	29
Khafre Pyramid	27
Great Pyramids of Giza	27
Siwa	23
Ramesseum	23
Colossoi of Memnon	22
Muizz_Street	20
Great_Hypostyle_Hall_of_Karnak	20
Bent_Pyramid	20
Hatshepsut	20
Temple_of_Kom_Ombo	19
Nabq_Protected_Area	19
Fatimid_Cemetery_in_Aswan	19
Saint_Catherine's_Monastery_Mount_Sinai	19
Colossal Statue of Ramesses II	19
Bayt_Al-Suhaymi	19
Ramessum	19
Goddess Isis with her child	19
Pyramid of Djoser	19

Insights from the error analysis of the initial model Same classes with different names (mistakes from the merging files) ex: Ramessum – Ramesseum Colossoi of Memnon – Colossi_of_memnon


```
misclassified_df["True Label"].value_counts().head(20)
```

True Label	count
Elephantine	29
Statue of Ramesses II	29
Khafre Pyramid	27
Great Pyramids of Giza	27
Siwa	23
Ramesseum	23
Colossoi of Memnon	22
Muizz_Street	20
Great_Hypostyle_Hall_of_Karnak	20
Bent_Pyramid	20
Hatshepsut	20
Temple_of_Kom_Ombo	19
Nabq_Protected_Area	19
Fatimid_Cemetery_in_Aswan	19
Saint_Catherine's_Monastery_Mount_Sinai	19
Colossal Statue of Ramesses II	19
Bayt_Al-Suhaymi	19
Ramessum	19
Goddess Isis with her child	19
Pyramid of Djoser	19

Also the top miss classified classes like siwa & Elephantine Have unrelated images in the class so it seems logical it make poor performance

error analysis cont

```python
misclassified_df.sample(30)
```

	Image Path	True Label	Predicted Label
564	unzipped_data/cleaned_data/Monastery_of_Saint_...	Monastery_of_Saint_Anthony	Kiosk_of_Qertassi
837	unzipped_data/cleaned_data/Islamic_Cairo/8.jpg	Islamic_Cairo	Great_Pyramid_of_Giza
988	unzipped_data/cleaned_data/Qubbet_el-Hawa/3.jpg	Qubbet_el-Hawa	Deir_el-Bahari
21	unzipped_data/cleaned_data/sphinx/82.jpg	sphinx	Giza_pyramid_complex
1948	unzipped_data/cleaned_data/Pyramid_of_Khafra/5...	Pyramid_of_Khafra	Khafre Pyramid
360	unzipped_data/cleaned_data/Colossoi of Memnon/...	Colossoi of Memnon	Colossi_of_Memnon
305	unzipped_data/cleaned_data/Khafre Pyramid/222.jpg	Khafre Pyramid	menkaure pyramid
1942	unzipped_data/cleaned_data/Ras_Muhammad/15.jpg	Ras_Muhammad	Ptolemaic_Temple_of_Hathor_in_Deir_el-Medina
233	unzipped_data/cleaned_data/Kitchener's_Island/...	Kitchener's_Island	Al-Manyal_Palace_Museum
1608	unzipped_data/cleaned_data/Al-Aqmar_Mosque/aug...	Al-Aqmar_Mosque	sphinx
1529	unzipped_data/cleaned_data/Ramessum/8 (1).jpg	Ramessum	Ramesseum
765	unzipped_data/cleaned_data/Temple_of_Khonsu_in...	Temple_of_Khonsu_in_Karnak	Tomb_of_Nakht_TT52
1752	unzipped_data/cleaned_data/Al-Aqmar_Mosque/aug...	Al-Aqmar_Mosque	Statue of King Zoser
898	unzipped_data/cleaned_data/Ramessum/49 (1).jpg	Ramessum	Ramesseum
1541	unzipped_data/cleaned_data/Pyramid_of_Djedefra...	Pyramid_of_Djedefra	Great_Sphinx_of_Giza
1888	unzipped_data/cleaned_data/Abdeen_Palace/aug_0...	Abdeen_Palace	Colossi_of_Memnon
1443	unzipped_data/cleaned_data/Esna_Temple/aug_10.jpg	Esna_Temple	Islamic_Cairo
1473	unzipped_data/cleaned_data/Garden_City,_Cairo/...	Garden_City,_Cairo	Soma_Bay
1922	unzipped_data/cleaned_data/Kiosk_of_Qertassi/a...	Kiosk_of_Qertassi	Colossoi of Memnon
1234	unzipped_data/cleaned_data/Bent_Pyramid/aug_57...	Bent_Pyramid	Edfu_Temple
1594	unzipped_data/cleaned_data/Red_Pyramid/13.jpg	Red_Pyramid	KV62
1748	unzipped_data/cleaned_data/Pyramid_of_Menkaure...	Pyramid_of_Menkaure	Al-Qurn
948	unzipped_data/cleaned_data/Siwa/23.jpg	Siwa	Edfu_Temple
673	unzipped_data/cleaned_data/Ptolemaic_Temple_of...	Ptolemaic_Temple_of_Hathor_in_Deir_el-Medina	Hatshepsut Statue
1668	unzipped_data/cleaned_data/Pyramid_of_Djoser/1...	Pyramid_of_Djoser	Pyramid of Djoser

ex_1

The samples from the misclassified classes provided valuable insights into the causes of the errors. In many cases, images from different classes were visually similar or mislabeled—for example, classifying a Sphinx image under the 'Giza_Pyramid_Complex' class. This suggests that some images, such as those of the Sphinx, may appear in multiple classes, leading to confusion during classification.

```python
same_classes = {
    "Ramessum" :  "Ramesseum",
    "Mosque of Al Mahmudiyah" : "Mosque_of_al-Mahmudiya",
    "Pyramid of Djoser" : "Pyramid_of_Djoser" ,
    "Colossoi of Memnon" : "Colossi_of_memnon",
    "Pyramid_of_Khafra" : "Khafre Pyramid",
     "Bibliotheca Alexandrina" : "Bibliotheca_Alexandrina",
     "Great_Hypostyle_Hall_of_Karnak" : "Great Hypostyle Hall of Karnak",
     "Bent pyramid for senefru" : "Bent_Pyramid",
    "sphinx" : "Great_Sphinx_of_Giza",
    "Isis with her child" : "Goddess Isis with her child",
    "Egyptian Museum Tahrir" : "Egyptian_Museum_(Cairo)",


}
classesToDrop = ["Mohandessin","siwa","Great_Pyramid_of_Giza","Pyramid_of_Khafra",
                 "Hatshepsut","Elephantine","Statue of Ramesses II","Giza_pyramid_complex",S"Giza_Plateau",
                 "Garden_City,_Cairo","Gezira","KV62","Mokattam","Mortuary_Temple_of_Hatshepsut",
                 "6_October_Bridge","Nabq_Protected_Area","Pyramid_of_Menkaure"]
```

ex_2

Final insights led us to merge certain overlapping classes and drop others that were either redundant or poorly represented.

error analysis cont

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```
for layer in base_model.layers:
    layer.trainable = False
```

```
model = Sequential([
    base_model,
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='softmax')
])
```

```
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=0.00001)
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
flatten_1 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 512)	12,845,568
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 178)	91,314

Total params: 27,651,570 (105.48 MB)
Trainable params: 12,936,882 (49.35 MB)
Non-trainable params: 14,714,688 (56.13 MB)

```
model.fit(train_gen, epochs=20, validation_data=test_gen, callbacks=[reduce_lr, early_stopping])
```

```
misclassified_df["True Label"].value_counts().head(20)
```

True Label	count
Abu_Haggag_Mosque	30
Montaza_Palace	30
Kom_el-Shoqafa	30
Cairo_Citadel	30
Ras_Muhammad	30
Qasr_al-Nil_Bridge	30
Abu_el-Abbas_el-Mursi_Mosque	30
Colossal Statue of Ramesses II	30
Qaed_Ibrahim_Mosque	30
Temple_of_Hibis	30
Egyptian_National_Military_Museum	30
Aqsunqur_Mosque	30
Bab_al-Nasr_(Cairo)	30
Theban_Necropolis	30
Wadi_Degla	30
Al-Ghuri_Complex	30
Giza_Zoo	30
Egyptian_Museum_(Cairo)	30
Temple_of_Seti_I_in_Abydos	30
Gebel_el-Silsila	30

dtype: int64

```
misclassified_df[(misclassified_df["True Label"]== "Egyptian_National_Military_Museum")|(misclassified_df["True Label"]=="")]
```

	Image Path	True Label	Predicted Label
95	unzipped_data/cleaned_data/Egyptian_National_M...	Egyptian_National_Military_Museum	Gayer-Anderson_Museum
353	unzipped_data/cleaned_data/Egyptian_National_M...	Egyptian_National_Military_Museum	Gayer-Anderson_Museum
499	unzipped_data/cleaned_data/Egyptian_National_M...	Egyptian_National_Military_Museum	Bayt_Al-Suhaymi

In the second error analysis we detect the weak classes and we collect more data to it

```
from icrawler.builtin import GoogleImageCrawler

classes = [
    "Khafre Pyramid",
     "Khan_el-Khalili", "Qaitbay Castle", "Na'ama Bay",
    "The Great Temple of Ramesses II", "Wadi_el-Raiyan", "Egyptian_National_Military_Museum",
    "Deir_el-Bahari", "Montaza_Palace", "Al-Azhar_Park_(Cairo)", "amenhotep iii and tiye",
    "Alexandria_Opera_House", "Nefertiti", "Agiba_beach", "Bent_Pyramid",
    "Great Hypostyle Hall of Karnak", "Qubbet_el-Hawa", "Ramses II Red Granite Statue",
    "Bagawat", "Fatimid_Cemetery_in_Aswan", "Temple_of_Hibis", "Luxor_Temple",
    "Pyramid_of_Sahure", "Great_Sphinx_of_Giza", "Islamic_Cairo",
    "Monastery_of_Saint_Simeon_in_Aswan", "Al-Ghuri_Complex", "Monastery_of_Saint_Macarius_the_Great",
    "Abu_el-Abbas_el-Mursi_Mosque", "Bayt_Al-Suhaymi", "Faiyum_Zoo", "bust of ramesses ii",
    "Sabil_of_Abd_al-Rahman_Katkhuda", "Babylon Fortress", "Theban_Necropolis",
    "Colossal Statue of Ramesses II", "Hatshepsut face", "Abu Haggag Mosque",
    "Pyramid_of_Djoser", "Madrasah_of_Al-Nasir_Muhammad", "Unfinished_obelisk_in_Aswan",
    "Ramesseum", "Ptolemaic Temple of Hathor in Deir el-Medina", "Qasr Qarun",
    "Saint George Church in Coptic Cairo", "White Monastery"
]

for name in classes:
    crawler = GoogleImageCrawler(storage={'root_dir': f'images/{name.replace(" ", "_")}'})
    crawler.crawl(keyword=name, max_num=35)
```

We collect 35 image for the weak classes from google and then we checked it manually that the image quality is good

Finallyyyyyy we got our final model

With accuracy 80%

Further Improvements & 90% Target

1

Advanced Augmentation

More diverse image transformations

2

Transfer Learning

Leverage pretrained networks for better features

3

Preprocessing & Data

Refined image preprocessing

Expand labeled dataset

Goal: Reach 90% classification accuracy

Recommendations & NLP Integration

NLP Use

Google Gemini API powers context-aware assistant

Functionality

Generates insightful artifact descriptions

Answers user queries with cultural accuracy

Recommendations

Feedback loops enhance chatbot accuracy

API & Web Overview

Gemini API Integration

llm_utils.py manages language model calls

Web Front End

A web interface was developed to allow users to easily interact with the model. Through a simple and user-friendly design, users can upload artifact images, receive AI-generated descriptions, and chat with the Horus AI assistant.

This frontend bridges the gap between complex AI models and real-world users, making the system accessible and practical.

Conclusion & Vision

Progress Summary

From data collection to 80% accuracy AI model

Importance

Uniting AI, NLP, and heritage education

Vision

Create interactive, smarter archaeology tools