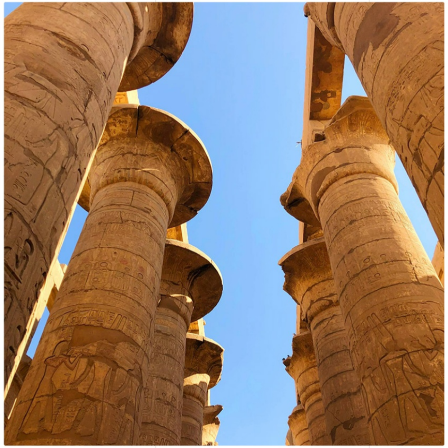
Transfer Learning Project: Hieroglyph Classification

Maram Mohamed Hanafy
Board Member

Project Introduction and Data Source

Started a transfer learning project focused on Hieroglyph Classification.

The main goal is to classify Egyptian hieroglyph symbols using transfer learning techniques.



Dataset sourced from Kaggle at the provided URL.

The dataset used is the Egyptian Hieroglyphs Dataset available on Kaggle.

Used URL with user credentials to verify authorized access and avoid spam.

Ensured access was verified by using credentials to confirm user authorization.

Data Loading Process

Downloaded and unzipped the dataset files.

The initial step involved obtaining the dataset and preparing it for use by extracting compressed files.

Imported necessary Python libraries for data processing.

Set up the programming environment by loading required libraries to handle and manipulate data.

Encountered an issue where images needed cropping before processing.

Discovered that raw images required cropping to ensure proper formatting and analysis.

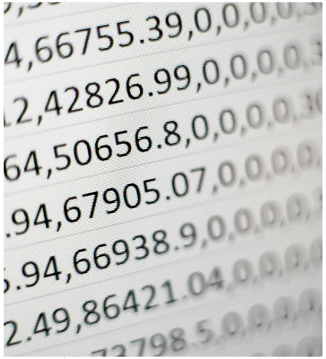
Loaded data using Python's os library to navigate directory structures.

Used os library functions to systematically access and organize files within folders.

Data Structure and Annotation Details

Dataset organized into three folders: train, test, and valid.

This structure helps separate data for training, testing, and validation purposes.



Each folder contains a CSV file with annotations.

The CSV files hold essential data for image processing tasks.

Annotations include x and y coordinates for bounding boxes.

Coordinates enable precise cropping of hieroglyph images.

Image labels for classification are included in annotations.

Labels support accurate categorization of hieroglyph images.

Class Discovery and Folder Verification

5 classes

Initial number of classes before verification

The dataset started with just 5 classes identified.

95 classes

Number of classes after folder verification

After checking image labels against folders, the classes increased significantly.

Image Cropping Method

Used bounding box coordinates from annotations to crop images.

Coordinates included x_min, x_max, y_min, and y_max for precise cropping around hieroglyphs.

Method allowed precise cropping around the hieroglyphs.

Ensured that the cropped images focused exactly on the regions of interest.

Post-cropping, TensorFlow accessed images with correct labels.

This enabled further processing and training using accurately labeled image data.

Preprocessing Steps

Images were cropped before resizing.

This initial step helped focus on the region of interest in the images.

Images were resized to a uniform size suitable for model input.

Ensured consistency in input dimensions for the neural network.

Preprocessing standardized image dimensions and formats.

This helped prepare the data effectively for transfer learning.

The resizing step was crucial for transfer learning preparation.

Standardizing images allows the pretrained model to work properly with the new dataset.

Model Selection: Why ResNet?

Dataset Contains Shape-Rich Symbols With Intricate Edges And Curves

The Hieroglyphs Have Detailed Visual Characteristics That Need Careful Capturing.

Residual Connections:

ResNet's Skip Connections Mitigate The Vanishing Gradient Problem, Making It Faster To Train

ResNet Is Highly Effective For Recognizing Detailed Visual Features In Hieroglyphs

This Makes It A Strong Base Model For Classification Of Detailed Symbols.

Model Preparation and Customization

Loaded the pretrained ResNet model and removed its top FC layers

Adapted the model to our dataset with 95 classes instead of ImageNet's 1000.

Set trainable = False on convolutional layers

Froze convolutional layers to prevent them from updating during training.

Tuned only the top fully connected layers

Optimized performance specifically for our classification task by training just the FC layers.

Training, Evaluation, and Visualization

Trained the customized model using the training dataset.

The model was trained specifically on the training data prepared for this project.

Evaluated the model on the testing dataset.

Model performance was assessed using a separate testing dataset to ensure unbiased evaluation.

Added a plotting function to visualize images alongside predicted classes.

This function helps in qualitatively assessing the model's accuracy by showing images with their predicted labels.

Visualization aids in qualitative assessment of model accuracy.

Seeing the predictions next to images provides insight into how well the model is performing beyond numeric metrics.