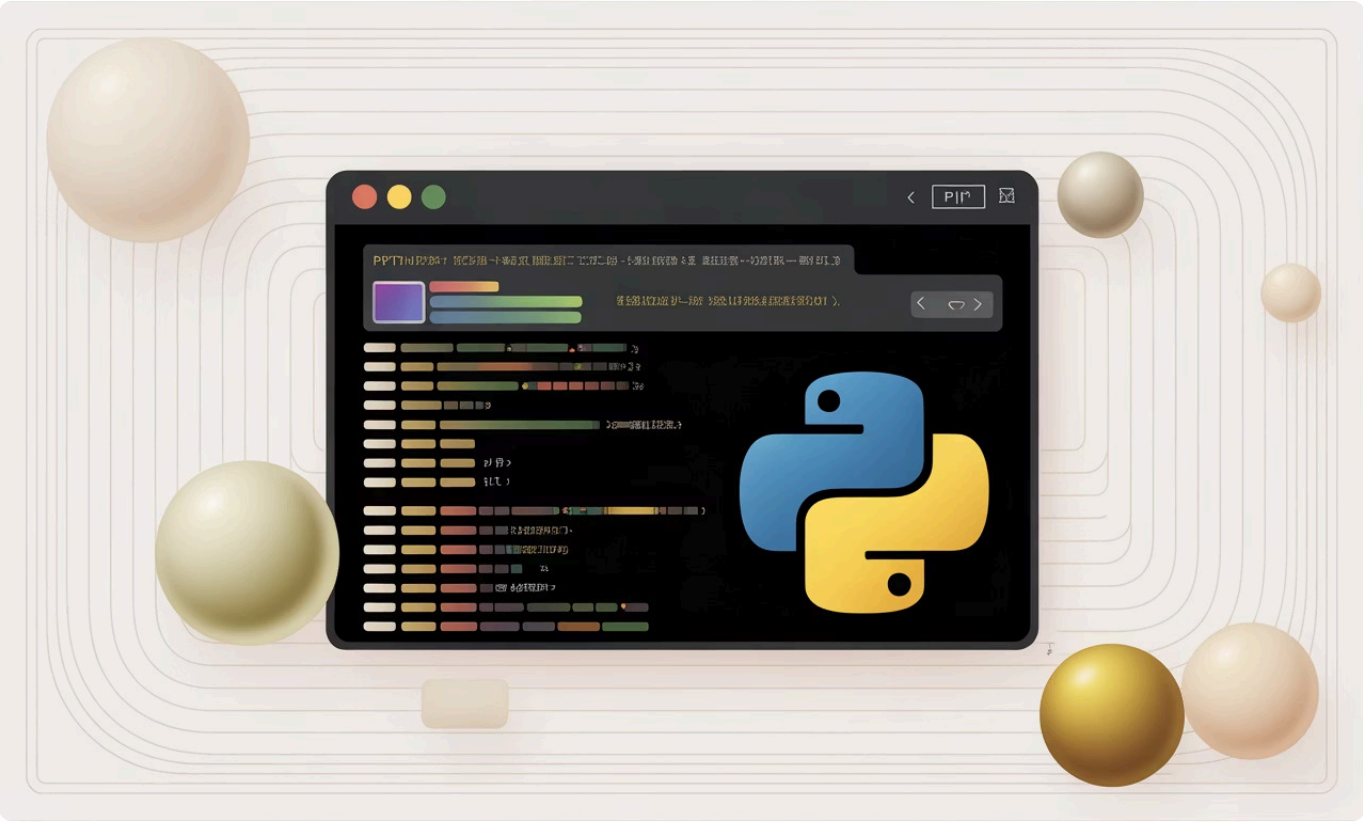
Image Dataset Processing in Google Colab

A complete guide to preparing and transforming image datasets for machine learning projects using Python libraries in a cloud environment.

Made with GAMMA

GETTING STARTED

Install Essential Dependencies



Required Libraries

Begin by installing the two core Python libraries needed for image processing and visualization. OpenCV handles image manipulation tasks, while Matplotlib enables visual inspection of your transformations.

```
!pip install opencv-python
!pip install matplotlib
```

These commands download and configure the necessary packages in your Colab environment.

Made with GAMMA

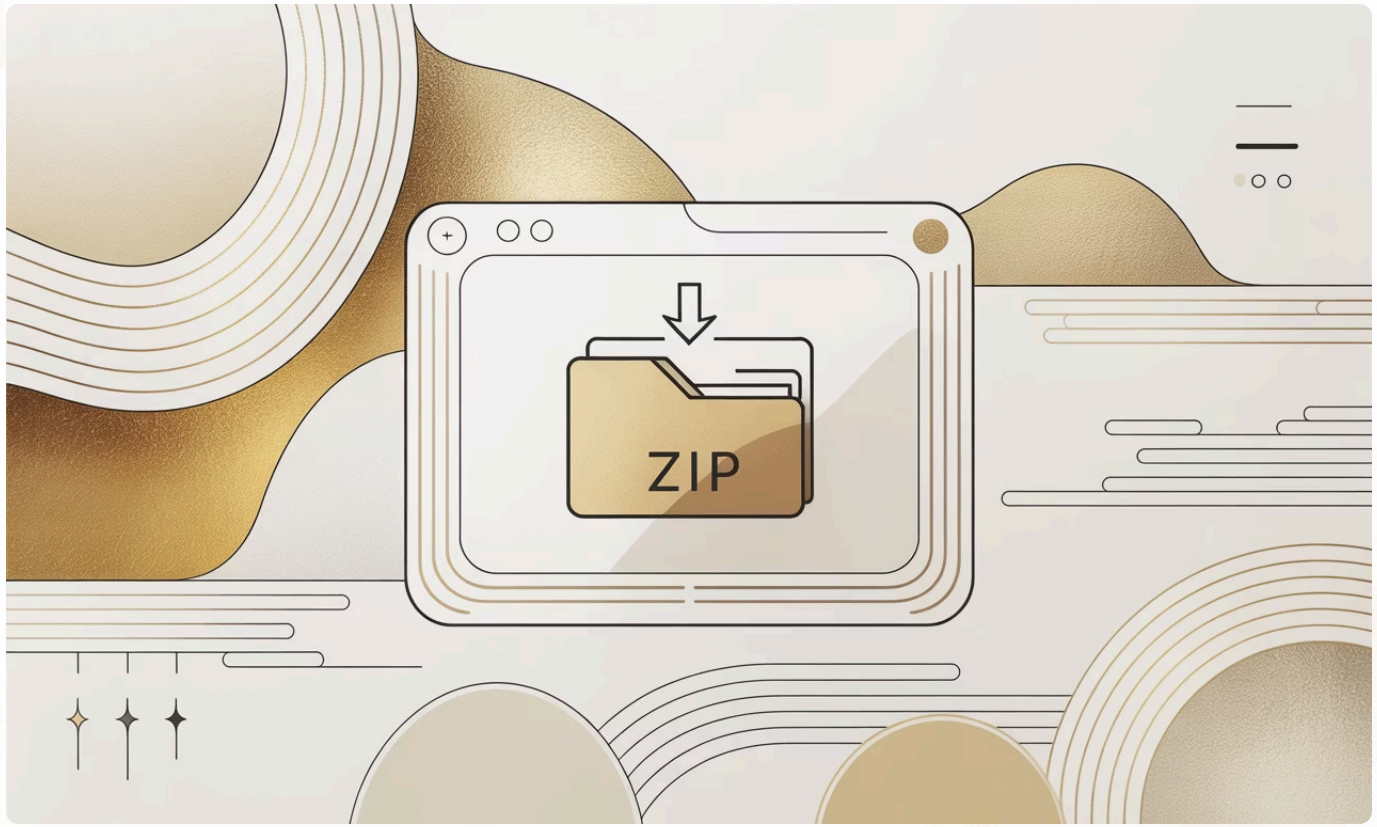
Upload Your Dataset to Colab

Interactive File Upload

Use Colab's built-in file upload widget to transfer your dataset from your local machine. The code prompts a file browser dialog, allowing you to select and upload your compressed dataset file.

```
from google.colab import files
uploaded = files.upload()
dataset_zip_path = list(uploaded.keys())[0]
```

This captures the uploaded filename automatically for the next processing step.



Made with GAMMA

DATA PREPARATION

Extract the Dataset Archive

Unpack your compressed dataset into an organized directory structure. The extraction process creates a dedicated folder and decompresses all images, making them accessible for processing operations.

```
import os
from zipfile import ZipFile

extract_path = "/content/dataset"
os.makedirs(extract_path, exist_ok=True)

with ZipFile(dataset_zip_path, 'r') as zip_ref:
 zip_ref.extractall(extract_path)

print(f"Dataset extracted to: {extract_path}")
```

Made with GAMMA

Configure Processing Parameters

Resize Dimensions

Width: 300px
Height: 300px

Standard square format for uniform dataset consistency

Rotation Angle

45 degrees

Adds augmentation variety to improve model robustness

Output Location

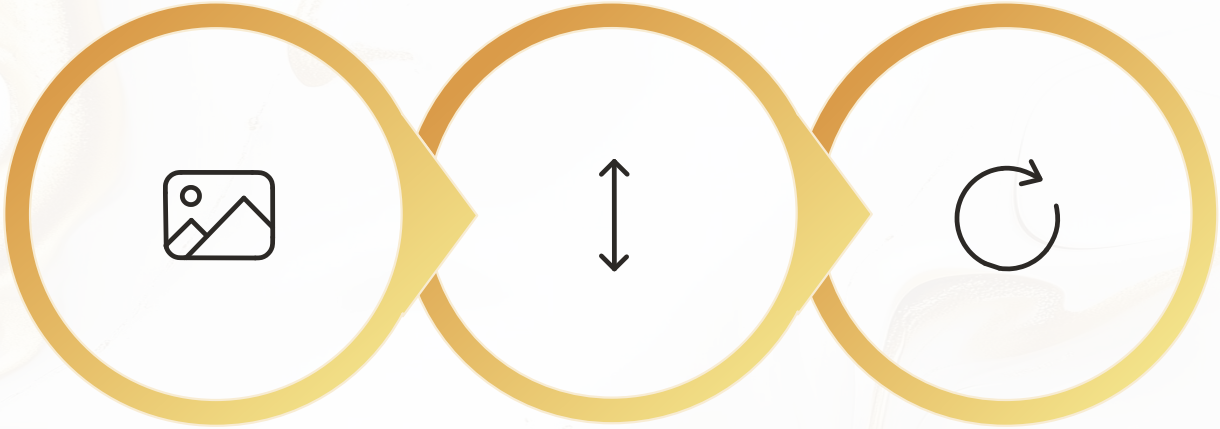
/content/processed_images

Dedicated folder for saving transformed results

```
resize_width = 300
resize_height = 300
rotation_angle = 45
output_folder = "/content/processed_images"
os.makedirs(output_folder, exist_ok=True)
```

Made with GAMMA

The Processing Pipeline



Load Image **Resize Image** **Rotate Image**

Each image undergoes a systematic transformation sequence. The pipeline reads images from your dataset, applies consistent resizing to standardize dimensions, and performs rotation augmentation to enhance dataset diversity.

Made with GAMMA

CORE IMPLEMENTATION

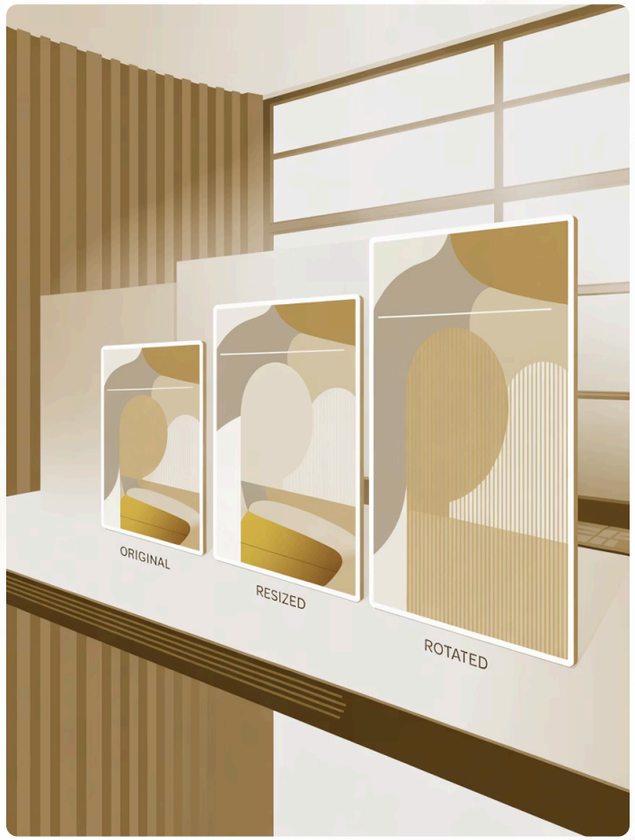
Complete Processing Code

This comprehensive script walks through your entire dataset directory, identifies image files, and applies both resize and rotation transformations while preserving the original files.

```
import cv2
import matplotlib.pyplot as plt
import os

for root, dirs, files in os.walk(extract_path):
 for file in files:
 if file.lower().endswith(('.jpg', '.jpeg', '.png')):
 img_path = os.path.join(root, file)
 img = cv2.imread(img_path)

if img is None:
continue

# Convert BGR to RGB for proper display
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Resize operation
resized_img = cv2.resize(img_rgb,
(resize_width, resize_height))

# Rotation operation
(h, w) = resized_img.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center,
rotation_angle, 1.0)
rotated_img = cv2.warpAffine(resized_img, M, (w, h))
```

Made with GAMMA

Visual Comparison Output



Side-by-Side Visualization

The code generates a three-panel display for each processed image, showing the original, resized, and rotated versions simultaneously. This visual feedback helps verify transformations are applied correctly.

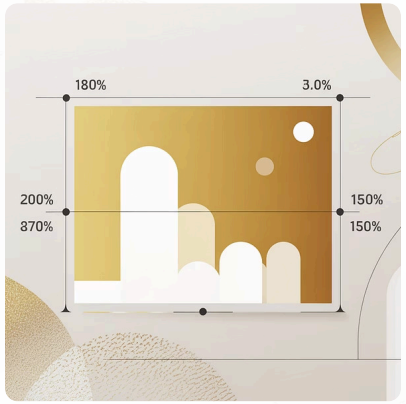
```
# Display all three images
plt.figure(figsize=(15,5))

plt.subplot(1,3,1)
plt.imshow(img_rgb)
plt.title("Original")
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(resized_img)
plt.title("Resized")
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(rotated_img)
plt.title("Rotated")
plt.axis('off')

plt.show()
```

Made with GAMMA

Key Processing Techniques

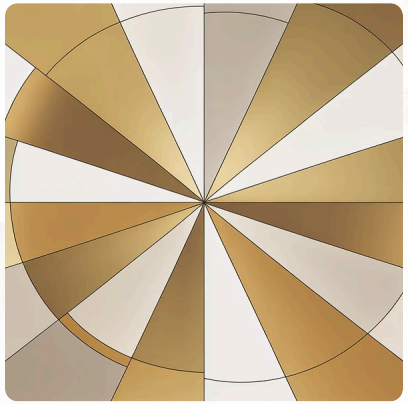
Color Space Conversion

OpenCV loads images in BGR format, but Matplotlib expects RGB. The conversion ensures accurate color display during visualization.



Uniform Resizing

Standardizes all images to consistent dimensions, ensuring compatibility with machine learning models that require fixed input sizes.



Center-Based Rotation

Applies geometric transformation using a rotation matrix calculated from the image center, preserving content while adding variation.

Made with GAMMA

Next Steps: Export Your Work

Optional: Download Results

Once processing is complete, you can package the transformed images into a ZIP archive for download to your local machine or transfer to cloud storage for model training.

Use Colab's file download functionality or integrate with Google Drive for seamless workflow continuation.

01

Verify outputs

Check visual results

02

Package files

Create ZIP archive

03

Download or sync

Transfer to storage

Made with GAMMA