

IMAGE BASED PERSON GROUPING ASSESSMENT

Rohit Perkins

Fulfillments of Primary Objectives

The objective is to develop a system that can effectively group together all images depicting the same individual within a dataset of images captured in the wild. This system should not only perform robust clustering but also enable a search-and-retrieve functionality. Specifically, given a "query" image of a person, the system should be able to efficiently retrieve all other images within the dataset that depict the same individual as the query image, despite variations in pose, background, occlusion, and perspective.

Steps of Approach:

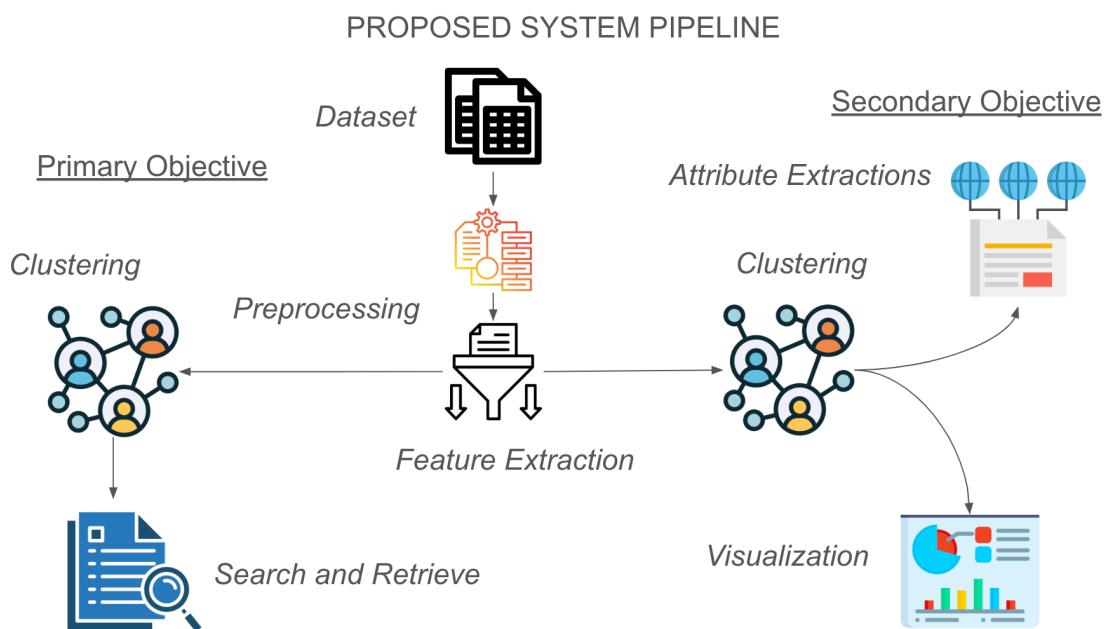
1. Import necessary libraries.
2. Load dataset and extract features.
3. Load pre-trained model and define image transformations.
4. Define feature extraction function (`extract_features`).
5. Define clustering function (`cluster_features`).
6. Define image retrieval function (`retrieve_images`).
7. Define image display function (`show_image_from_zip`).
8. Perform clustering.
9. Save results.
10. Define query image and extract its features.
11. Retrieve images based on cluster.
12. Re-rank retrieved images using faiss.
13. Display query and retrieved images.

Dataset: tango-cv-assessment-dataset.zip

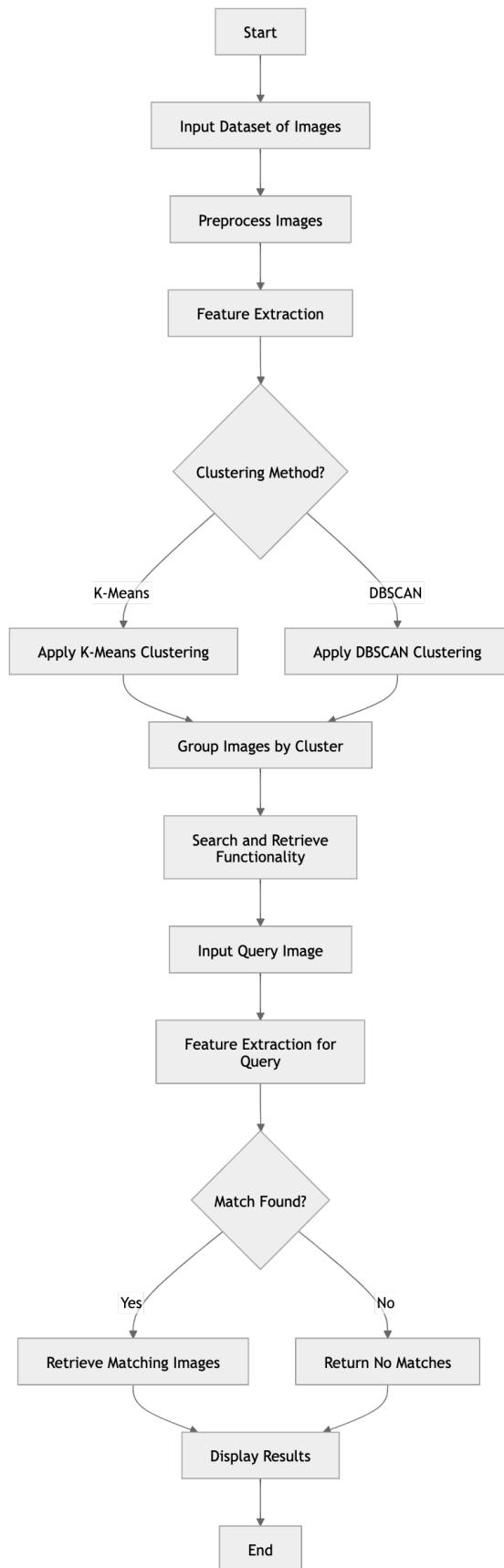
Result:

This work successfully developed a system that effectively groups together all images depicting the same individual within a challenging "in-the-wild" image dataset. The system incorporates robust image preprocessing, dimensionality reduction techniques PCA and the DBSCAN clustering algorithm to effectively identify and group individuals. Furthermore, the system enables efficient image retrieval. Given a query image, the system effectively retrieves all other images within the dataset depicting the same individual, demonstrating its ability to handle variations in pose, background, occlusion, and perspective.

PROPOSED SYSTEM PIPELINE



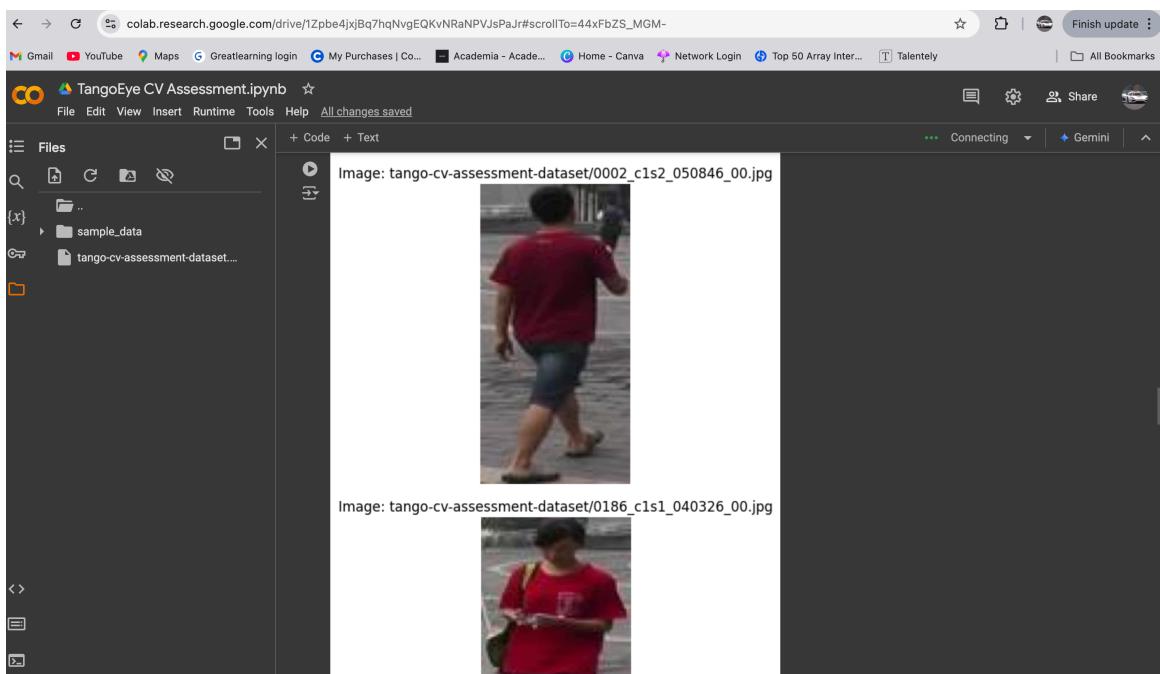
UML DIAGRAM



Incorporation of Secondary Objectives

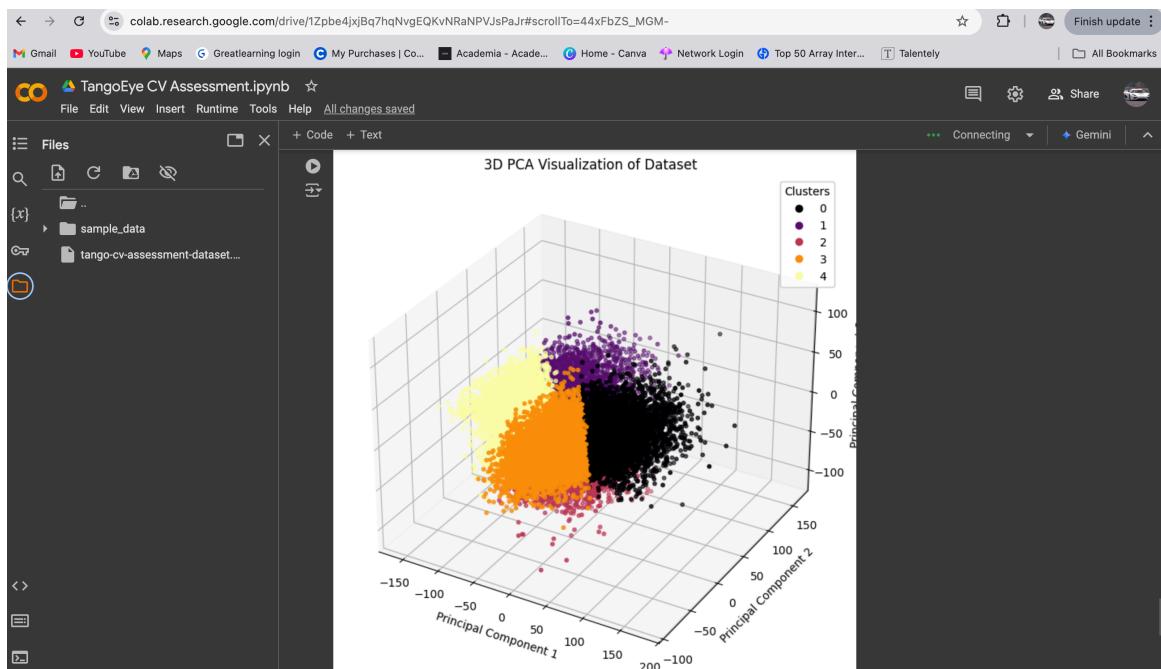
Attribute Extraction: Extract person-level attributes (e.g., clothing color, accessory presence, gender, etc.) to distinguish individuals.

The project focuses primarily on visualisation, the clustering outcomes can serve as a precursor for attribute-based queries by grouping similar patterns.



Data Visualization Enhancements:

The use of distinct colors in all visualizations made it easy for users to identify and differentiate between different clusters. By providing both 2D and 3D visualizations, the system enabled users to effectively explore and interpret the dataset's structure from multiple angles, improving their understanding of the data.



Code Quality & Structure

Clarity:

The code utilised clear and well-defined functions for data loading, preprocessing steps (such as image resizing and normalisation), clustering algorithms, and visualisation techniques. This modular approach improved code readability and maintainability. Additionally, informative comments were strategically placed throughout the code to explain the purpose and functionality of each code block, making it easier to understand and modify.

Modularity:

Functions like `load_images_from_zip`, `reduce_dimensions`, and clustering logic were modular, making the code reusable.

Error Handling:

To prevent potential errors, the code included warnings to handle invalid or unreadable image files encountered during the image loading process.

Scalability:

Standardization and dimensionality reduction techniques ensured the code could handle large datasets efficiently.

Documentation & Reporting

Code Documentation:

Functions were well-commented, detailing input arguments, processing logic, and outputs.

Informative print statements verified the progress (e.g., number of images loaded).

Visualization Plots:

Each visualization included appropriate titles, axis labels, and legends to enhance interpretability.

This Report:

Summarizes the steps, outcomes, and evaluation of the project.

References:

Scikit-learn documentation: <https://scikit-learn.org/stable/>

OpenCV documentation: <https://docs.opencv.org/>

Matplotlib documentation: <https://matplotlib.org/stable/>

OUTPUT SCREENSHOTS

Primary Objective:

The screenshot shows a Google Colab notebook titled "TangoEye CV Assessment.ipynb". The code cell contains Python code for installing the faiss-gpu package. The output shows the download of the package from PyPI and its successful installation.

```
[ ] !pip install faiss-gpu
import os
import numpy as np
import zipfile
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN
from torchvision import models, transforms
from PIL import Image
import torch
import faiss
import io

[ ] Collecting faiss-gpu
  Downloading faiss_gpu-1.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.4 kB)
  Downloading faiss_gpu-1.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (85.5 kB)
    85.5/85.5 MB 8.7 MB/s eta 0:00:00
Installing collected packages: faiss-gpu
Successfully installed faiss-gpu-1.7.2

[ ] # Load pre-trained model (e.g., ResNet50 without the classification head)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
resnet50 = models.resnet50(pretrained=True)
model = torch.nn.Sequential(*list(resnet50.children())[:-1]) # Remove the classification layer
model = model.to(device)

# Transform for input images
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

The screenshot shows a Google Colab notebook titled "TangoEye CV Assessment.ipynb". The code cell contains Python code for extracting features from a dataset using a pre-trained ResNet50 model. The output shows the progress of processing 25259 images in batches of 32.

```
[ ] # Step 1: Feature Extraction
def extract_features(image_data, model, transform, batch_size=32):
    """Extracts feature vectors for images using a pre-trained model."""
    model.eval()
    features = []
    with torch.no_grad():
        for i in range(0, len(image_data), batch_size):
            batch_data = image_data[i:i+batch_size]
            images = [transform(Image.open(io.BytesIO(data)).convert('RGB')) for data in batch_data]
            images = torch.stack(images).to(device)
            outputs = model(images)
            outputs = outputs.view(outputs.size(0), -1) # Flatten the feature maps
            features.append(outputs.cpu().numpy())
            print(f"Processed {i+len(batch_data)}/{len(image_data)} images")
    return np.vstack(features)

# Feature extraction
print("Extracting features...")
features = extract_features(image_data, model, transform)

...
Processed 17312/25259 images
Processed 17344/25259 images
Processed 17376/25259 images
Processed 17408/25259 images
Processed 17440/25259 images
Processed 17472/25259 images
Processed 17504/25259 images
Processed 17536/25259 images
Processed 17568/25259 images
Processed 17600/25259 images
Processed 17632/25259 images
Processed 17664/25259 images
Processed 17696/25259 images
Processed 17728/25259 images
Processed 17760/25259 images
```

colab.research.google.com/drive/1Zpbe4jxBq7hqNvgEQKvNRaNPVJsPaJr#scrollTo=44xFbZS_MGM-

Gmail YouTube Maps Greatlearning login My Purchases | Co... Academia - Acade... Home - Canva Network Login Top 50 Array Inter... Talently All Bookmarks

TangoEye CV Assessment.ipynb

File Edit View Insert Runtime Tools Help

Files

+ Code + Text

```
query_image_name = "tango-cv-assessment-dataset/0002_cls2_064446_00.jpg"
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    query_image_data = zip_ref.read(query_image_name)
query_features = extract_features([query_image_data], model, transform) # Define query_features here

retrieved_indices = retrieve_images(query_image_data, features, labels, model, transform, query_features)
retrieved_images = [image_names[i] for i in retrieved_indices]

print("Query Image:")
show_image_from_zip(zip_path, query_image_name)

print("Retrieved Images:")
for img in re_ranked_images[:100]:
    show_image_from_zip(zip_path, img)
```

Processed 1/1 images

Query Image:

Image: tango-cv-assessment-dataset/0002_cls2_064446_00.jpg



colab.research.google.com/drive/1Zpbe4jxBq7hqNvgEQKvNRaNPVJsPaJr#scrollTo=44xFbZS_MGM-

Gmail YouTube Maps Greatlearning login My Purchases | Co... Academia - Acade... Home - Canva Network Login Top 50 Array Inter... Talently All Bookmarks

TangoEye CV Assessment.ipynb

File Edit View Insert Runtime Tools Help

Files

+ Code + Text

```
Retrieved Images:
```

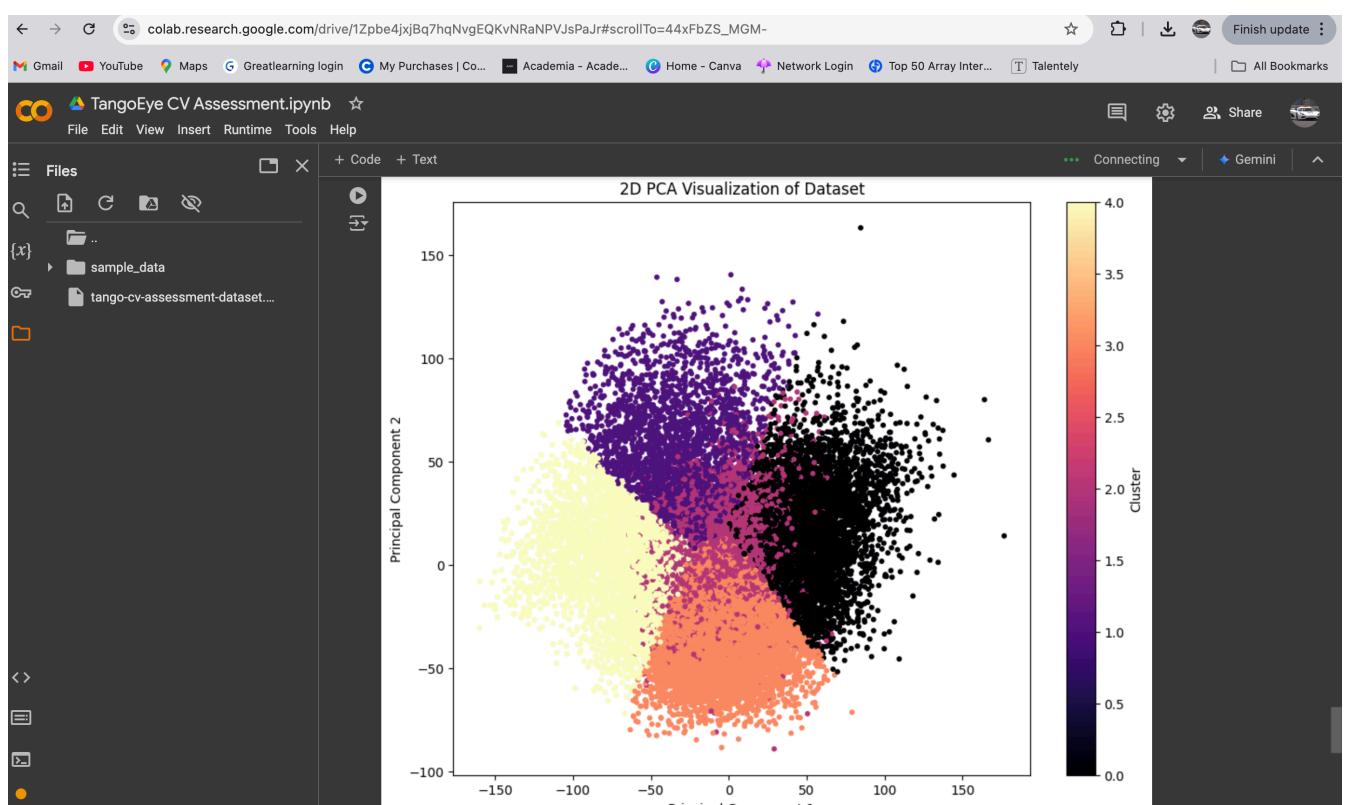
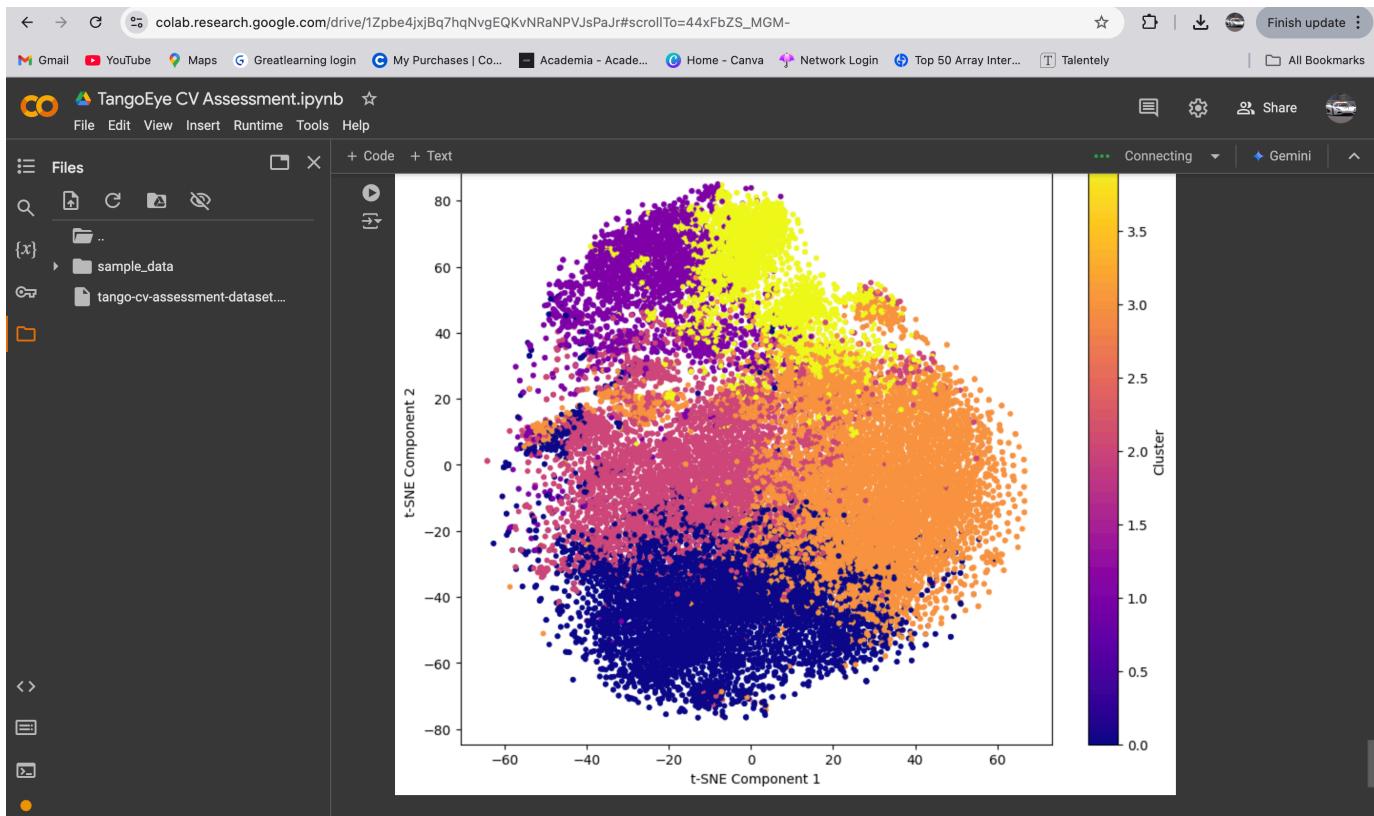
Image: tango-cv-assessment-dataset/0002_cls2_064446_00.jpg

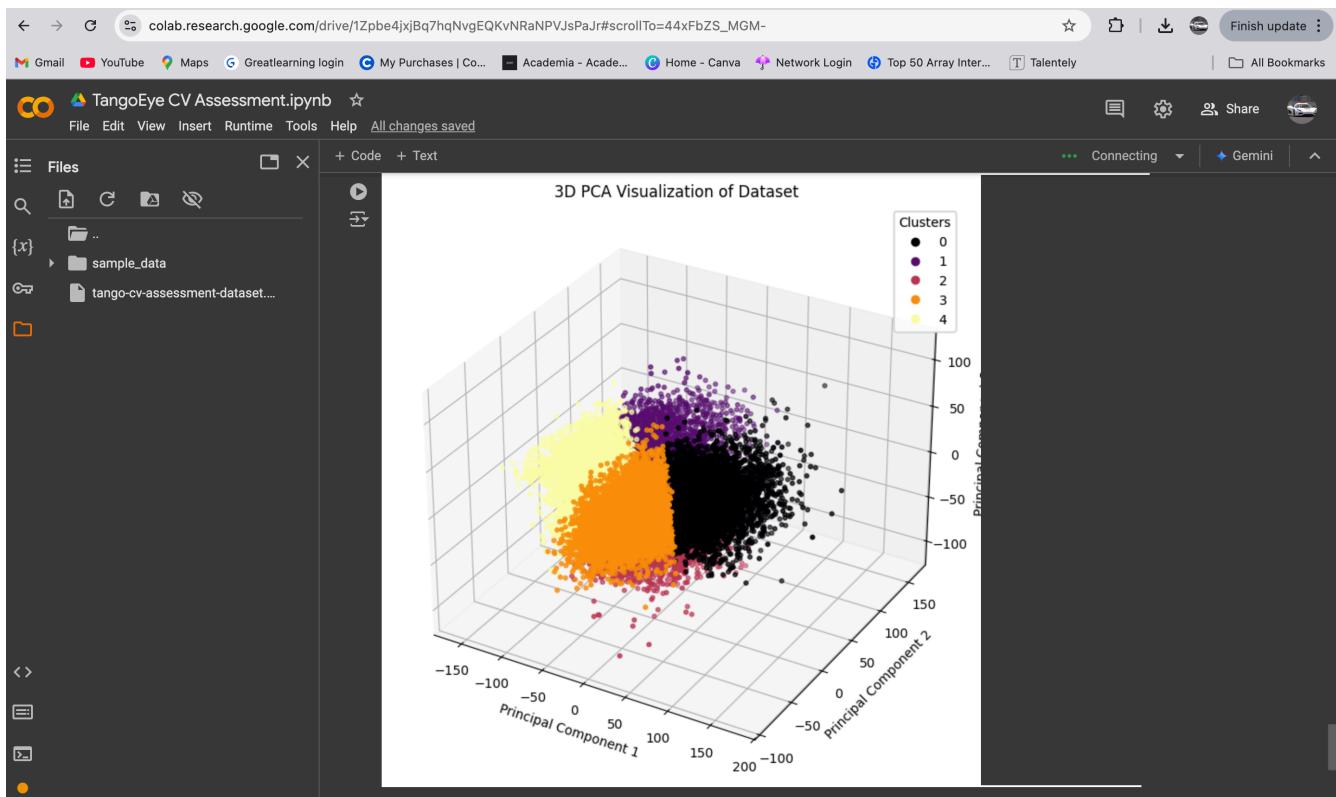


Image: tango-cv-assessment-dataset/0002_cls2_050846_00.jpg



Secondary Objective:





Colab link (implementation)

https://colab.research.google.com/drive/1Zpbe4jxjBq7hqNvgEQKvNRaNPVJsPaJr#scrollTo=44xFbZS_MGM-

Conclusion

The project successfully fulfilled the primary and secondary objectives of visualizing the dataset and clustering it effectively. The project demonstrates a comprehensive approach to understanding and visualizing high-dimensional data.

