**Install and Download Required Files**

# Install required package google download

Install 'gdown' package

# Download dataset files using Google Drive IDs

Download file with ID '1iaHvLiKck4GS6CVmK6WId-LvqaUDzsUu' # Train and validation image pairs

Download file with ID '159N7-SLmNHomvc60JB3s1rvJMkvPxylF' # Annotations

# Unzip downloaded dataset files

Unzip 'Train-Val-DS.zip' to '/content/dataset/' # Train and validation images

Unzip 'train-val-annotations.zip' to '/content/dataset/' # Annotations

**Load Necessary Libraries**

# Load required libraries

Import 'json', 'pandas', 'numpy', 'PIL.Image', 'sklearn.preprocessing.LabelEncoder'

Import 'tensorflow', 'tensorflow.keras.models.Sequential'

Import 'tensorflow.keras.layers.Conv2D, MaxPooling2D, Flatten, Dense, Dropout'

Import 'matplotlib.pyplot'

**Load JSON Annotations**

# Function to load JSON files

Define function 'load\_json(file\_path)':

Open file in read mode

Load JSON data

Return JSON data

# Load training and validation annotations

Load class\_train = 'class\_train\_annotation.json'

Load class\_val = 'class\_val\_annotation.json'

Load capt\_train = 'capt\_train\_annotations.json'

Load capt\_val = 'capt\_val\_annotations.json'

**Prepare Training and Validation Data**

# Prepare Training Data

Initialize empty list 'train\_data'

For each 'item' in class\_train:

Extract 'pair\_id'

Extract first available 'label'

Find corresponding 'caption' from capt\_train

Extract 'image1' and 'image2' names

Append data to 'train\_data'

Convert 'train\_data' to Pandas DataFrame 'train\_df'

# Prepare Validation Data (same process as training)

Initialize empty list 'val\_data'

For each 'item' in class\_val:

Extract 'pair\_id', 'label', 'caption', 'image1', 'image2'

Append data to 'val\_data'

Convert 'val\_data' to Pandas DataFrame 'val\_df'

**Encode Labels for Model Training**

# Encode labels using LabelEncoder

Initialize LabelEncoder

Transform train labels

Transform validation labels

**Load and Preprocess Image Pairs**

# Function to load and combine image pairs

Define function 'load\_image\_pair(image1\_name, image2\_name)':

Open images

Resize images to (128, 128)

Convert to NumPy array and normalize

Concatenate images along last axis (forming 6 channels)

Return combined image

# Load images for training and validation

Convert all image pairs in train\_df and val\_df using 'load\_image\_pair'

Store as NumPy arrays 'X\_train' and 'X\_val'

# Convert labels to one-hot encoding

Convert 'train\_labels' and 'val\_labels' to categorical format

**Define and Compile CNN Model**

# Define CNN Model

Initialize Sequential Model

Add Conv2D layers with activation 'relu'

Add MaxPooling layers

Add Flatten layer

Add Dense layer with activation 'relu'

Add Dropout layer

Add Output layer with activation 'softmax'

Compile model with 'adam' optimizer and categorical cross-entropy loss

Print model summary

**Train the CNN Model**

# Train Model

Fit model on training data for 10 epochs with batch size 64

Validate using validation data

**Plot Training and Validation Accuracy**

# Plot Training History

Plot training and validation accuracy over epochs

**Predict Subtle Differences Using the Model**

# Function to predict subtle differences

Define function 'predict\_difference(image1\_name, image2\_name)':

Load combined image using 'load\_image\_pair'

Make a prediction using trained model

Return predicted class label

# Example prediction

Predict difference for test images

**Save the Trained Model**

# Save model

Save trained model as 'my\_cnn\_model.h5'

**Load Multiple Annotation Files**

# Function to load multiple annotation files

Define function 'load\_annotations\_from\_multiple\_files(file\_paths)':

Initialize empty list 'annotations'

For each file path:

Open file

Load JSON data

Extend annotations list

Return annotations

# Load all annotation files

Define annotation file paths

Call 'load\_annotations\_from\_multiple\_files'

**Predict All Differences (Shape,Color,Texture) in image pairs**

# Function to predict all differences in images

Define function 'predict\_all\_differences(image1\_name, image2\_name, annotations)':

Initialize dictionary for predictions

For each annotation in annotations:

If 'image1\_name' matches, extract attributes

Store detected attributes (color, shape, texture)

Return predictions

# Example prediction for an image pair

Predict differences for 'v3\_1034\_after' and 'v3\_1034\_before'

**Display Predictions with Images**

# Function to display predictions with images

Define function 'display\_full\_prediction(image1\_name, image2\_name, annotations)':

Open images

Get predictions using 'predict\_all\_differences'

Display images and prediction results in a formatted table

# Example usage of display function

Call 'display\_full\_prediction' for an example image pair