

Data Collection and Preprocessing Phase

Date	29 November 2024
Team ID	739984
Project Title	AI-Generated vs. Real Image Classifier Using Deep Learning.
Maximum Marks	6 Marks

Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	Description
Data Overview	Use a dataset of AI-generated and real images, categorized by themes like objects, landscapes, and faces. Sources include Kaggle and proprietary datasets.
Resizing	Resize all images to a fixed size (e.g., 224x224) to ensure compatibility with deep learning models like CNNs.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Use algorithms like Canny or Sobel to emphasize edges, potentially enhancing the differentiation between real and AI-generated images.




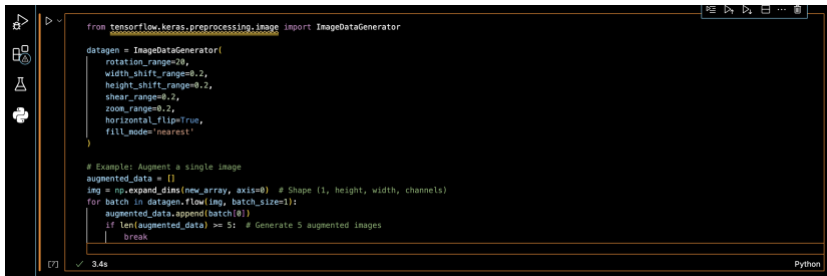
Color Space Conversion	Experiment with converting RGB images to grayscale, HSV, or LAB color spaces to explore the impact on classification accuracy.
Image Cropping	Crop images to focus on the regions containing objects of interest.
Batch Normalization	Integrate batch normalization layers in the neural network to speed up training and enhance the model's performance.
Data Preprocessing Code Screenshots	
Loading Data	 <pre>data = "/Users/shivanjintoppula/Desktop/nikhilal/train" categories = ['AlGenerated', 'Real']</pre>
Resizing	 <pre>img_size = 48</pre>
Normalization	 <pre>try: img_array = cv.imread(img_path) if img_array is None: print(f"Skipping unreadable file: {img}") continue new_array = cv.resize(img_array, (img_size, img_size)) new_array = new_array / 255.0 training_data.append([new_array, classes])</pre>
Data Augmentation	 <pre>from tensorflow.keras.preprocessing.image import ImageDataGenerator datagen = ImageDataGenerator(rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest') # Example: Augment a single image augmented_data = [] img = cv.imread(img_path) # Shape (1, height, width, channels) for batch in datagen.flow(img, batch_size=1): augmented_data.append(batch[0]) if len(augmented_data) == 5: # Generate 5 augmented images break</pre>
Denoising	NO
Edge Detection	NO
Color Space Conversion	NO

Image Cropping

```

X_train = []
y_train = []

for features, label in training_data:
    X_train.append(features)
    y_train.append(label)

X_train = np.array(X_train).reshape(-1, img_size, img_size, 3)
y_train = np.array(y_train)

```

[10] ✓ 0.0s Python

Batch Normalization

```

model = keras.Sequential([
    keras.layers.Conv2D(32, (3,3), activation='relu', input_shape = (48,48,3)),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Dropout(0.2),

    keras.layers.Conv2D(64, (3,3), activation='relu'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Dropout(0.2),

    keras.layers.Conv2D(128, (3,3), activation='relu'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Dropout(0.2),

    keras.layers.Conv2D(256, (3,3), activation='relu'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Dropout(0.2),

    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

```

[10] ✓ 0.0s Python