CS 203: Software Tools & Techniques for AI

IIT Gandhinagar

Sem-II - 2024-25

LAB 05

GROUP : 2

Pranav Somase 22110200

Om Gupta 22110174

Task 1: Data Augmentation

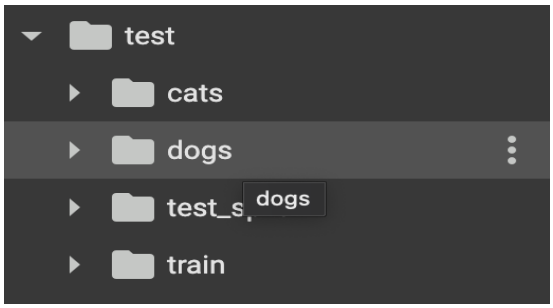
- Download the Cat & Dog Dataset (https://www.kaggle.com/datasets/samuelcortinhas/cats-and-dogs-image-classification?select=test) (Only download the test dataset; do not take the training dataset)

Create a train and test set (train-test ratio should be 80:20%).

CODE :

```
# Define paths
dataset_path = "/content/drive/MyDrive/test"  # Change if needed
train_path = "/content/drive/MyDrive/test/train"
test_path = "/content/drive/MyDrive/test/test_split"

# Create train and test directories
os.makedirs(train_path, exist_ok=True)
os.makedirs(test_path, exist_ok=True)

# Get all image file paths
cat_images = [os.path.join(dataset_path, "cats", img) for img in os.listdir(os.path.join(dataset_path, "cats"))]
dog_images = [os.path.join(dataset_path, "dogs", img) for img in os.listdir(os.path.join(dataset_path, "dogs"))]

# Split into train (80%) and test (20%)
cat_train, cat_test = train_test_split(cat_images, test_size=0.2, random_state=42)
dog_train, dog_test = train_test_split(dog_images, test_size=0.2, random_state=42)

# Create category-specific folders in train and test sets
for folder in ["cats", "dogs"]:
    os.makedirs(os.path.join(train_path, folder), exist_ok=True)
    os.makedirs(os.path.join(test_path, folder), exist_ok=True)

# Function to copy images instead of moving them
def copy_images(file_list, dest_folder):
    for file in file_list:
        shutil.copy(file, os.path.join(dest_folder, os.path.basename(file)))

# Copy images to train and test directories
copy_images(cat_train, os.path.join(train_path, "cats"))
copy_images(dog_train, os.path.join(train_path, "dogs"))
copy_images(cat_test, os.path.join(test_path, "cats"))
copy_images(dog_test, os.path.join(test_path, "dogs"))

print("Dataset successfully split into train and test sets without modifying the original images.")
```

```
Dataset successfully split into train and test sets without modifying the original images.
```



test_split→ contains the test split (20% of the downloaded dataset)
train→ contains the train split (80% of the downloaded dataset, before augmentation)

Create Custom Function using Augly, which will perform multiple random data augmentations according to input. (At least 10 data augmentation needs to be added like rotate, cropping, blur …)

CODE :

```python
def apply_random_augmentations(image_path, num_augmentations=3):
    """
    Apply multiple random augmentations from AugLy to an image.
    Parameters:
    - image_path: Path to the input image
    - num_augmentations: Number of random augmentations to apply (default = 3)
    Returns:
    - Augmented image (PIL.Image object)
    """
    # Load the image
    img = PIL.Image.open(image_path).convert("RGB")

    # Define a list of available functional augmentations
    augmentations = [
        lambda x: imaugs.rotate(x, degrees=random.uniform(-30, 30)),  # Rotation
        lambda x: imaugs.crop(x, x1=0.15, y1=0.15, x2=0.85, y2=0.85),  # Cropping
        lambda x: imaugs.blur(x, radius=random.uniform(1, 3)),  # Blur
        lambda x: imaugs.color_jitter(x, brightness_factor=1.2, contrast_factor=1.2, saturation_factor=1.2),
        lambda x: imaugs.grayscale(x, mode="luminosity"),  # Convert to Grayscale
        lambda x: imaugs.brightness(x, factor=random.uniform(0.7, 1.3)),  # Brightness
        lambda x: imaugs.shuffle_pixels(x, factor=random.uniform(0.1, 0.5)),  # Pixel Shuffle (New)
        lambda x: imaugs.hflip(x),  # Horizontal Flip
        lambda x: imaugs.contrast(x, factor=random.uniform(0.5, 1.5)),  # Contrast Adjustment
        lambda x: imaugs.change_aspect_ratio(x, ratio=random.uniform(0.8, 1.2)),  # Aspect Ratio Change
    ]

    # Randomly choose augmentations to apply
    chosen_augmentations = random.sample(augmentations, num_augmentations)

    # Apply augmentations sequentially
    for aug in chosen_augmentations:
        img = aug(img)

    return img
```

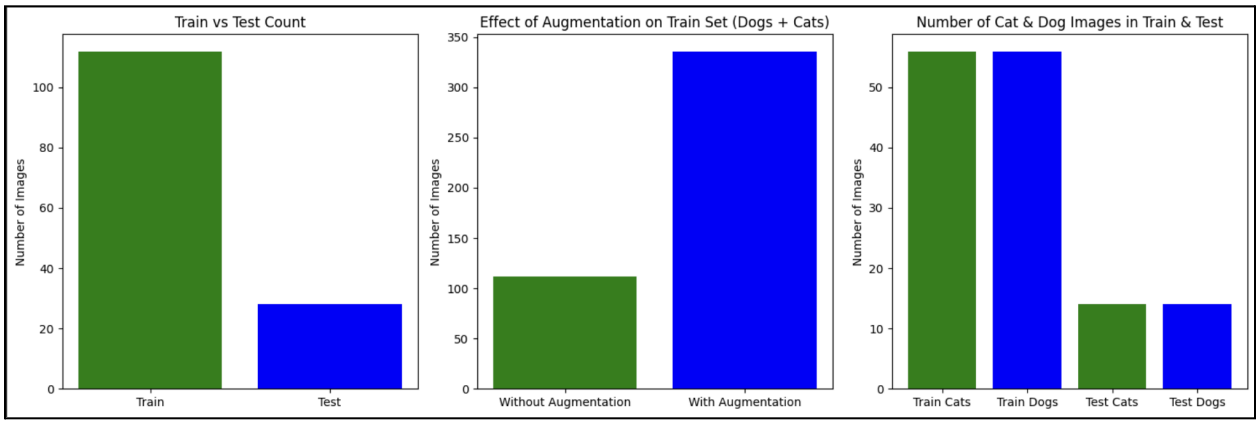
Perform data augmentation using the above function, only on the train set. (The number of augmented images should be twice the train set, and images should be augmented thrice example: cropped → rotate→Blur) (Second augmentation should be different)

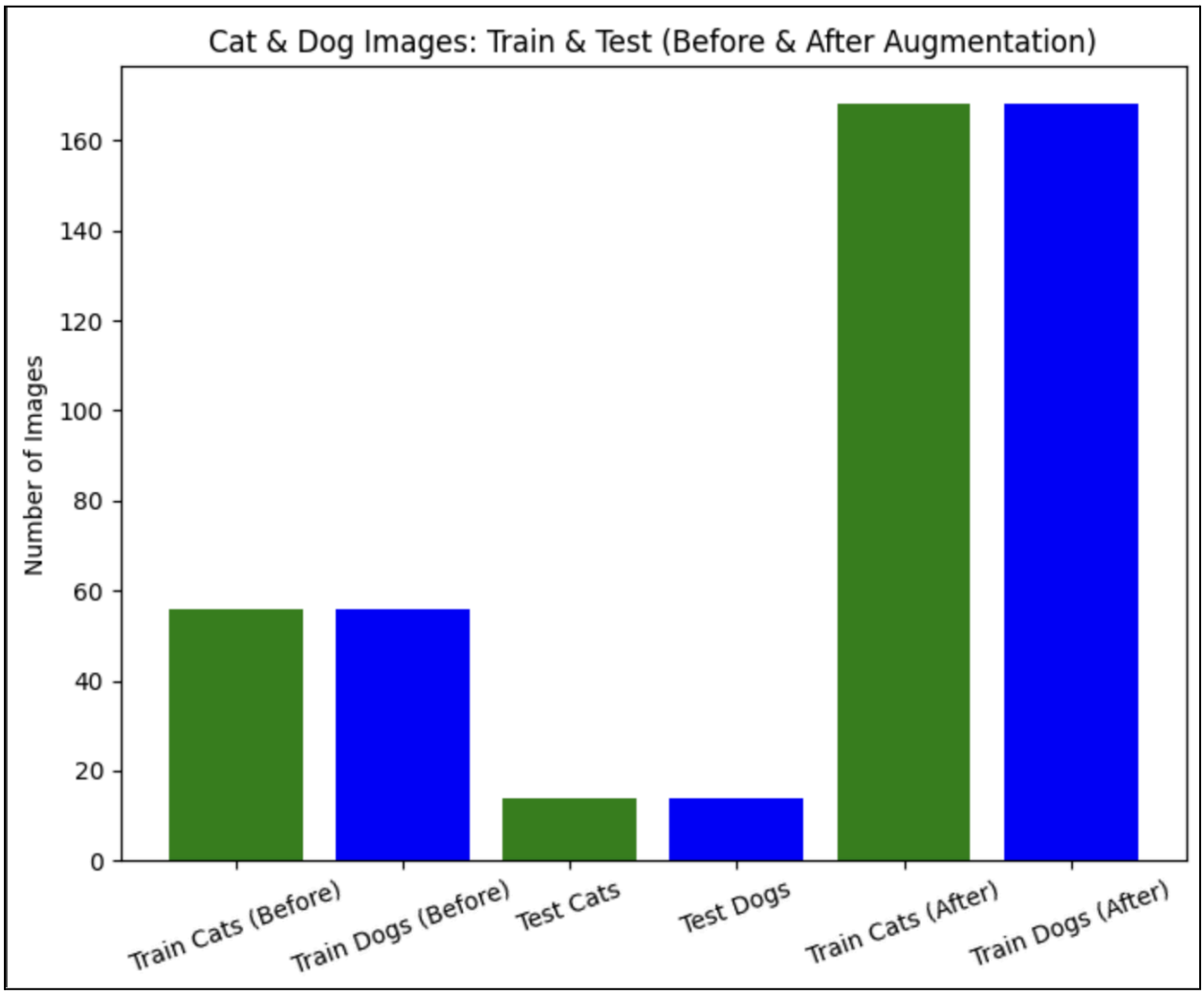
CODE :

```
# Define original paths
train_dir = "/content/drive/MyDrive/test/train"  # Original train folder
test_dir = "/content/drive/MyDrive/test/test_split"  # Test folder
aug_train_dir = "/content/drive/MyDrive/test/train_augmented"  # Augmented train folder

categories = ["cats", "dogs"]

# Function to copy the train folder
def copy_train_folder(src, dest):
    if os.path.exists(dest):
        shutil.rmtree(dest)  # Remove existing copy if any
    shutil.copytree(src, dest)
    print(f"Copied train dataset to {dest}")

# Function to save augmented images
def save_augmented_image(image, save_path, filename, count):
    new_filename = f"{os.path.splitext(filename)[0]}_aug{count}.jpg"
    image.save(os.path.join(save_path, new_filename))

# Copy the train dataset before augmentation
copy_train_folder(train_dir, aug_train_dir)

# Get dataset statistics before augmentation
original_train_counts = {cat: len(os.listdir(os.path.join(aug_train_dir, cat))) for cat in categories}
original_test_counts = {cat: len(os.listdir(os.path.join(test_dir, cat))) for cat in categories}

print(f"Original Train Set Count (Copied): {original_train_counts}")
print(f"Original Test Set Count: {original_test_counts}")
```

```
# Augment images in the copied train folder
for category in categories:
    cat_path = os.path.join(aug_train_dir, category)
    images = [img for img in os.listdir(cat_path) if img.lower().endswith(('png', 'jpg', 'jpeg'))]

    for image_name in images:
        image_path = os.path.join(cat_path, image_name)

        # Apply augmentation twice with different transformations
        aug_img1 = apply_random_augmentations(image_path, num_augmentations=3)
        aug_img2 = apply_random_augmentations(image_path, num_augmentations=3)

        # Save the augmented images
        save_augmented_image(aug_img1, cat_path, image_name, 1)
        save_augmented_image(aug_img2, cat_path, image_name, 2)

print("Augmentation completed on copied dataset.")
```

Show the statistics of the newly created dataset. (Old dataset count and new dataset count)





Task 2: Model Training

Choose (microsoft/resnet-50)model from the hugging face and initialize its new weights :

ResNet-50 is a **deep convolutional neural network (CNN)** that introduced **residual learning** to solve the vanishing gradient problem in very deep networks. It consists of **50 layers**.

Initial Layers:

- **7×7 Convolution** (64 filters, stride 2) → **BatchNorm + ReLU**
- **3×3 Max Pooling** (stride 2)

Residual Blocks (Bottleneck Architecture):

- **Stage 1:** 3 residual blocks
- **Stage 2:** 4 residual blocks
- **Stage 3:** 6 residual blocks
- **Stage 4:** 3 residual blocks
  (Each block consists of three convolutions: 1×1, 3×3, 1×1)

Final Layers:

- **Global Average Pooling**
- **Fully Connected Layer** (1000 classes for ImageNet)

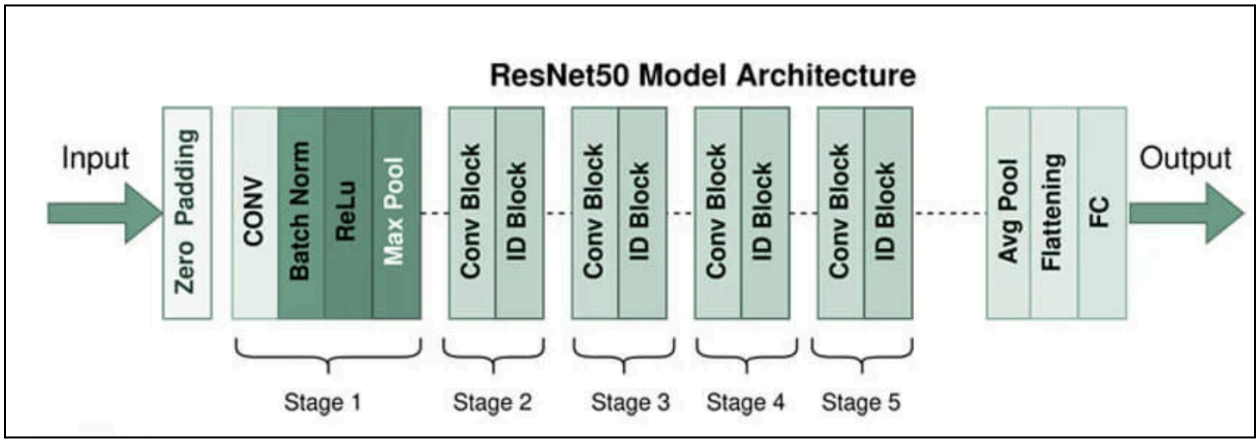


Image source : https://www.educba.com/keras-resnet50/

```
# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load model with new random weights
model_name = "microsoft/resnet-50"
feature_extractor = AutoFeatureExtractor.from_pretrained(model_name)
model = AutoModelForImageClassification.from_pretrained(model_name, num_labels=2, ignore_mismatched_sizes=True)

# Save initial model weights for consistency
initial_state_dict = model.state_dict()
```

Model Parameters for Non - Augmented Dataset :

```
Initial Model Parameters (Before Training on Non-Augmented Dataset):
resnet.embedder.embedder.convolution.weight: 0.001238
resnet.embedder.embedder.normalization.weight: 1.259472
resnet.embedder.embedder.normalization.bias: 3.082108
resnet.encoder.stages.0.layers.0.shortcut.convolution.weight: -0.016100
resnet.encoder.stages.0.layers.0.shortcut.normalization.weight: 2.095687
resnet.encoder.stages.0.layers.0.shortcut.normalization.bias: -0.179410
resnet.encoder.stages.0.layers.0.layer.0.convolution.weight: -0.012449
resnet.encoder.stages.0.layers.0.layer.0.normalization.weight: 0.847122
resnet.encoder.stages.0.layers.0.layer.0.normalization.bias: 0.287815
resnet.encoder.stages.0.layers.0.layer.1.convolution.weight: 0.000854
resnet.encoder.stages.0.layers.0.layer.1.normalization.weight: 0.830128
resnet.encoder.stages.0.layers.0.layer.1.normalization.bias: 1.118425
resnet.encoder.stages.0.layers.0.layer.2.convolution.weight: -0.000388
resnet.encoder.stages.0.layers.0.layer.2.normalization.weight: 0.092829
resnet.encoder.stages.0.layers.0.layer.2.normalization.bias: -0.179386
resnet.encoder.stages.0.layers.1.layer.0.convolution.weight: -0.006094
resnet.encoder.stages.0.layers.1.layer.0.normalization.weight: 0.906537
resnet.encoder.stages.0.layers.1.layer.0.normalization.bias: -0.361555
resnet.encoder.stages.0.layers.1.layer.1.convolution.weight: -0.004041
resnet.encoder.stages.0.layers.1.layer.1.normalization.weight: 1.344903
resnet.encoder.stages.0.layers.1.layer.1.normalization.bias: 0.418416
resnet.encoder.stages.0.layers.1.layer.2.convolution.weight: -0.003594
resnet.encoder.stages.0.layers.1.layer.2.normalization.weight: 0.070339
resnet.encoder.stages.0.layers.1.layer.2.normalization.bias: -0.227762
resnet.encoder.stages.0.layers.2.layer.0.convolution.weight: -0.013760
resnet.encoder.stages.0.layers.2.layer.0.normalization.weight: 1.756795
resnet.encoder.stages.0.layers.2.layer.0.normalization.bias: -0.322390
resnet.encoder.stages.0.layers.2.layer.1.convolution.weight: -0.006002
resnet.encoder.stages.0.layers.2.layer.1.normalization.weight: 1.868877
resnet.encoder.stages.0.layers.2.layer.1.normalization.bias: 0.458595
resnet.encoder.stages.0.layers.2.layer.2.convolution.weight: -0.002394
resnet.encoder.stages.0.layers.2.layer.2.normalization.weight: 0.024615
resnet.encoder.stages.0.layers.2.layer.2.normalization.bias: -0.254909
```

```
resnet.encoder.stages.1.layers.0.shortcut.convolution.weight: -0.012940
resnet.encoder.stages.1.layers.0.shortcut.normalization.weight: 1.833669
resnet.encoder.stages.1.layers.0.shortcut.normalization.bias: 0.578247
resnet.encoder.stages.1.layers.0.layer.0.convolution.weight: -0.027263
resnet.encoder.stages.1.layers.0.layer.0.normalization.weight: 2.108437
resnet.encoder.stages.1.layers.0.layer.0.normalization.bias: -1.007985
resnet.encoder.stages.1.layers.0.layer.1.convolution.weight: -0.004321
resnet.encoder.stages.1.layers.0.layer.1.normalization.weight: 1.993834
resnet.encoder.stages.1.layers.0.layer.1.normalization.bias: 1.161901
resnet.encoder.stages.1.layers.0.layer.2.convolution.weight: 0.000002
resnet.encoder.stages.1.layers.0.layer.2.normalization.weight: 0.008543
resnet.encoder.stages.1.layers.0.layer.2.normalization.bias: -0.957221
resnet.encoder.stages.1.layers.1.layer.0.convolution.weight: -0.013665
resnet.encoder.stages.1.layers.1.layer.0.normalization.weight: 1.382959
resnet.encoder.stages.1.layers.1.layer.0.normalization.bias: -0.560060
resnet.encoder.stages.1.layers.1.layer.1.convolution.weight: -0.010007
resnet.encoder.stages.1.layers.1.layer.1.normalization.weight: 1.674074
resnet.encoder.stages.1.layers.1.layer.1.normalization.bias: -0.425525
resnet.encoder.stages.1.layers.1.layer.2.convolution.weight: 0.000823
resnet.encoder.stages.1.layers.1.layer.2.normalization.weight: -0.103106
resnet.encoder.stages.1.layers.1.layer.2.normalization.bias: -1.086348
resnet.encoder.stages.1.layers.2.layer.0.convolution.weight: -0.017445
resnet.encoder.stages.1.layers.2.layer.0.normalization.weight: 1.278715
resnet.encoder.stages.1.layers.2.layer.0.normalization.bias: -0.561450
resnet.encoder.stages.1.layers.2.layer.1.convolution.weight: -0.008267
resnet.encoder.stages.1.layers.2.layer.1.normalization.weight: 2.056053
resnet.encoder.stages.1.layers.2.layer.1.normalization.bias: -0.235906
resnet.encoder.stages.1.layers.2.layer.2.convolution.weight: -0.000188
resnet.encoder.stages.1.layers.2.layer.2.normalization.weight: -0.105416
resnet.encoder.stages.1.layers.2.layer.2.normalization.bias: -0.780575
resnet.encoder.stages.1.layers.3.layer.0.convolution.weight: -0.015356
resnet.encoder.stages.1.layers.3.layer.0.normalization.weight: 1.799552
resnet.encoder.stages.1.layers.3.layer.0.normalization.bias: -1.113798
resnet.encoder.stages.1.layers.3.layer.1.convolution.weight: -0.008691
resnet.encoder.stages.1.layers.3.layer.1.normalization.weight: 1.754685
resnet.encoder.stages.1.layers.3.layer.1.normalization.bias: -0.441345
```

```
resnet.encoder.stages.1.layers.3.layer.2.normalization.bias: -0.980411
resnet.encoder.stages.2.layers.0.shortcut.convolution.weight: -0.008258
resnet.encoder.stages.2.layers.0.shortcut.normalization.weight: 1.764252
resnet.encoder.stages.2.layers.0.shortcut.normalization.bias: 0.187652
resnet.encoder.stages.2.layers.0.layer.0.convolution.weight: -0.011994
resnet.encoder.stages.2.layers.0.layer.0.normalization.weight: 2.024277
resnet.encoder.stages.2.layers.0.layer.0.normalization.bias: -1.499428
resnet.encoder.stages.2.layers.0.layer.1.convolution.weight: -0.002500
resnet.encoder.stages.2.layers.0.layer.1.normalization.weight: 2.020979
resnet.encoder.stages.2.layers.0.layer.1.normalization.bias: 0.905397
resnet.encoder.stages.2.layers.0.layer.2.convolution.weight: -0.000426
resnet.encoder.stages.2.layers.0.layer.2.normalization.weight: 0.061592
resnet.encoder.stages.2.layers.0.layer.2.normalization.bias: -1.657230
resnet.encoder.stages.2.layers.1.layer.0.convolution.weight: -0.011037
resnet.encoder.stages.2.layers.1.layer.0.normalization.weight: 1.947607
resnet.encoder.stages.2.layers.1.layer.0.normalization.bias: -0.728212
resnet.encoder.stages.2.layers.1.layer.1.convolution.weight: -0.008194
resnet.encoder.stages.2.layers.1.layer.1.normalization.weight: 1.985618
resnet.encoder.stages.2.layers.1.layer.1.normalization.bias: -0.807819
resnet.encoder.stages.2.layers.1.layer.2.convolution.weight: -0.000149
resnet.encoder.stages.2.layers.1.layer.2.normalization.weight: -0.033727
resnet.encoder.stages.2.layers.1.layer.2.normalization.bias: -1.123693
resnet.encoder.stages.2.layers.2.layer.0.convolution.weight: -0.012251
resnet.encoder.stages.2.layers.2.layer.0.normalization.weight: 1.973441
resnet.encoder.stages.2.layers.2.layer.0.normalization.bias: -1.203886
resnet.encoder.stages.2.layers.2.layer.1.convolution.weight: -0.009476
resnet.encoder.stages.2.layers.2.layer.1.normalization.weight: 1.634989
resnet.encoder.stages.2.layers.2.layer.1.normalization.bias: -0.933010
resnet.encoder.stages.2.layers.2.layer.2.convolution.weight: 0.000887
resnet.encoder.stages.2.layers.2.layer.2.normalization.weight: -0.031106
resnet.encoder.stages.2.layers.2.layer.2.normalization.bias: -0.945812
resnet.encoder.stages.2.layers.3.layer.0.convolution.weight: -0.012816
resnet.encoder.stages.2.layers.3.layer.0.normalization.weight: 1.896779
resnet.encoder.stages.2.layers.3.layer.0.normalization.bias: -1.462643
resnet.encoder.stages.2.layers.3.layer.1.convolution.weight: -0.009570
resnet.encoder.stages.2.layers.3.layer.1.normalization.weight: 1.810677
```

```
resnet.encoder.stages.2.layers.3.layer.1.convolution.weight: -0.009570
resnet.encoder.stages.2.layers.3.layer.1.normalization.weight: 1.810677
resnet.encoder.stages.2.layers.3.layer.1.normalization.bias: -0.800505
resnet.encoder.stages.2.layers.3.layer.2.convolution.weight: -0.000114
resnet.encoder.stages.2.layers.3.layer.2.normalization.weight: -0.055987
resnet.encoder.stages.2.layers.3.layer.2.normalization.bias: -1.390908
resnet.encoder.stages.2.layers.4.layer.0.convolution.weight: -0.011289
resnet.encoder.stages.2.layers.4.layer.0.normalization.weight: 1.904067
resnet.encoder.stages.2.layers.4.layer.0.normalization.bias: -1.611392
resnet.encoder.stages.2.layers.4.layer.1.convolution.weight: -0.009701
resnet.encoder.stages.2.layers.4.layer.1.normalization.weight: 1.933307
resnet.encoder.stages.2.layers.4.layer.1.normalization.bias: -0.423970
resnet.encoder.stages.2.layers.4.layer.2.convolution.weight: -0.000748
resnet.encoder.stages.2.layers.4.layer.2.normalization.weight: 0.009316
resnet.encoder.stages.2.layers.4.layer.2.normalization.bias: -1.628630
resnet.encoder.stages.2.layers.5.layer.0.convolution.weight: -0.013736
resnet.encoder.stages.2.layers.5.layer.0.normalization.weight: 1.785921
resnet.encoder.stages.2.layers.5.layer.0.normalization.bias: -1.842249
resnet.encoder.stages.2.layers.5.layer.1.convolution.weight: -0.008854
resnet.encoder.stages.2.layers.5.layer.1.normalization.weight: 1.909714
resnet.encoder.stages.2.layers.5.layer.1.normalization.bias: -0.283483
resnet.encoder.stages.2.layers.5.layer.2.convolution.weight: -0.000262
resnet.encoder.stages.2.layers.5.layer.2.normalization.weight: 0.031181
resnet.encoder.stages.2.layers.5.layer.2.normalization.bias: -1.398192
resnet.encoder.stages.3.layers.0.shortcut.convolution.weight: -0.018376
resnet.encoder.stages.3.layers.0.shortcut.normalization.weight: 1.529910
resnet.encoder.stages.3.layers.0.shortcut.normalization.bias: -1.791760
resnet.encoder.stages.3.layers.0.layer.0.convolution.weight: -0.013306
resnet.encoder.stages.3.layers.0.layer.0.normalization.weight: 1.770316
resnet.encoder.stages.3.layers.0.layer.0.normalization.bias: -2.061581
resnet.encoder.stages.3.layers.0.layer.1.convolution.weight: -0.010335
resnet.encoder.stages.3.layers.0.layer.1.normalization.weight: 2.157323
resnet.encoder.stages.3.layers.0.layer.1.normalization.bias: -0.604424
resnet.encoder.stages.3.layers.0.layer.2.convolution.weight: -0.000847
resnet.encoder.stages.3.layers.0.layer.2.normalization.weight: 0.041416
resnet.encoder.stages.3.layers.0.layer.2.normalization.bias: -1.972201
```

```
resnet.encoder.stages.3.layers.1.layer.0.convolution.weight: -0.021780
resnet.encoder.stages.3.layers.1.layer.0.normalization.weight: 2.100202
resnet.encoder.stages.3.layers.1.layer.0.normalization.bias: -1.166298
resnet.encoder.stages.3.layers.1.layer.1.convolution.weight: -0.016312
resnet.encoder.stages.3.layers.1.layer.1.normalization.weight: 1.772130
resnet.encoder.stages.3.layers.1.layer.1.normalization.bias: -1.446008
resnet.encoder.stages.3.layers.1.layer.2.convolution.weight: 0.000835
resnet.encoder.stages.3.layers.1.layer.2.normalization.weight: -0.043142
resnet.encoder.stages.3.layers.1.layer.2.normalization.bias: -1.786957
resnet.encoder.stages.3.layers.2.layer.0.convolution.weight: -0.029455
resnet.encoder.stages.3.layers.2.layer.0.normalization.weight: 1.824968
resnet.encoder.stages.3.layers.2.layer.0.normalization.bias: -1.627085
resnet.encoder.stages.3.layers.2.layer.1.convolution.weight: -0.018025
resnet.encoder.stages.3.layers.2.layer.1.normalization.weight: 1.655676
resnet.encoder.stages.3.layers.2.layer.1.normalization.bias: -1.362353
resnet.encoder.stages.3.layers.2.layer.2.convolution.weight: -0.000674
resnet.encoder.stages.3.layers.2.layer.2.normalization.weight: 0.003714
resnet.encoder.stages.3.layers.2.layer.2.normalization.bias: -1.391422
classifier.1.weight: -0.000336
classifier.1.bias: -0.001342
```

Model Parameters for Augmented Dataset :

```
Model Parameters Before Training on Augmented Dataset (Should Match Initial):
resnet.embedder.embedder.convolution.weight: 0.001228
resnet.embedder.embedder.normalization.weight: 1.259522
resnet.embedder.embedder.normalization.bias: 3.081957
resnet.encoder.stages.0.layers.0.shortcut.convolution.weight: -0.016097
resnet.encoder.stages.0.layers.0.shortcut.normalization.weight: 2.095673
resnet.encoder.stages.0.layers.0.shortcut.normalization.bias: -0.179424
resnet.encoder.stages.0.layers.0.layer.0.convolution.weight: -0.012448
resnet.encoder.stages.0.layers.0.layer.0.normalization.weight: 0.847279
resnet.encoder.stages.0.layers.0.layer.0.normalization.bias: 0.288008
resnet.encoder.stages.0.layers.0.layer.1.convolution.weight: 0.000848
resnet.encoder.stages.0.layers.0.layer.1.normalization.weight: 0.830143
resnet.encoder.stages.0.layers.0.layer.1.normalization.bias: 1.118321
resnet.encoder.stages.0.layers.0.layer.2.convolution.weight: -0.000392
resnet.encoder.stages.0.layers.0.layer.2.normalization.weight: 0.092841
resnet.encoder.stages.0.layers.0.layer.2.normalization.bias: -0.179400
resnet.encoder.stages.0.layers.1.layer.0.convolution.weight: -0.006104
resnet.encoder.stages.0.layers.1.layer.0.normalization.weight: 0.906631
resnet.encoder.stages.0.layers.1.layer.0.normalization.bias: -0.361468
resnet.encoder.stages.0.layers.1.layer.1.convolution.weight: -0.004072
resnet.encoder.stages.0.layers.1.layer.1.normalization.weight: 1.344912
resnet.encoder.stages.0.layers.1.layer.1.normalization.bias: 0.418408
resnet.encoder.stages.0.layers.1.layer.2.convolution.weight: -0.003589
resnet.encoder.stages.0.layers.1.layer.2.normalization.weight: 0.070442
resnet.encoder.stages.0.layers.1.layer.2.normalization.bias: -0.227786
resnet.encoder.stages.0.layers.2.layer.0.convolution.weight: -0.013771
resnet.encoder.stages.0.layers.2.layer.0.normalization.weight: 1.756845
resnet.encoder.stages.0.layers.2.layer.0.normalization.bias: -0.322313
resnet.encoder.stages.0.layers.2.layer.1.convolution.weight: -0.006019
resnet.encoder.stages.0.layers.2.layer.1.normalization.weight: 1.868889
resnet.encoder.stages.0.layers.2.layer.1.normalization.bias: 0.458566
resnet.encoder.stages.0.layers.2.layer.2.convolution.weight: -0.002410
resnet.encoder.stages.0.layers.2.layer.2.normalization.weight: 0.024683
resnet.encoder.stages.0.layers.2.layer.2.normalization.bias: -0.254920
```

```
resnet.encoder.stages.1.layers.0.shortcut.convolution.weight: -0.012948
resnet.encoder.stages.1.layers.0.shortcut.normalization.weight: 1.833670
resnet.encoder.stages.1.layers.0.shortcut.normalization.bias: 0.578297
resnet.encoder.stages.1.layers.0.layer.0.convolution.weight: -0.027259
resnet.encoder.stages.1.layers.0.layer.0.normalization.weight: 2.108483
resnet.encoder.stages.1.layers.0.layer.0.normalization.bias: -1.007924
resnet.encoder.stages.1.layers.0.layer.1.convolution.weight: -0.004314
resnet.encoder.stages.1.layers.0.layer.1.normalization.weight: 1.993883
resnet.encoder.stages.1.layers.0.layer.1.normalization.bias: 1.161878
resnet.encoder.stages.1.layers.0.layer.2.convolution.weight: -0.000012
resnet.encoder.stages.1.layers.0.layer.2.normalization.weight: 0.008538
resnet.encoder.stages.1.layers.0.layer.2.normalization.bias: -0.957171
resnet.encoder.stages.1.layers.1.layer.0.convolution.weight: -0.013663
resnet.encoder.stages.1.layers.1.layer.0.normalization.weight: 1.382981
resnet.encoder.stages.1.layers.1.layer.0.normalization.bias: -0.560125
resnet.encoder.stages.1.layers.1.layer.1.convolution.weight: -0.010009
resnet.encoder.stages.1.layers.1.layer.1.normalization.weight: 1.674082
resnet.encoder.stages.1.layers.1.layer.1.normalization.bias: -0.425529
resnet.encoder.stages.1.layers.1.layer.2.convolution.weight: 0.000825
resnet.encoder.stages.1.layers.1.layer.2.normalization.weight: -0.103090
resnet.encoder.stages.1.layers.1.layer.2.normalization.bias: -1.086365
resnet.encoder.stages.1.layers.2.layer.0.convolution.weight: -0.017446
resnet.encoder.stages.1.layers.2.layer.0.normalization.weight: 1.278775
resnet.encoder.stages.1.layers.2.layer.0.normalization.bias: -0.561387
resnet.encoder.stages.1.layers.2.layer.1.convolution.weight: -0.008275
resnet.encoder.stages.1.layers.2.layer.1.normalization.weight: 2.056045
resnet.encoder.stages.1.layers.2.layer.1.normalization.bias: -0.235914
resnet.encoder.stages.1.layers.2.layer.2.convolution.weight: -0.000179
resnet.encoder.stages.1.layers.2.layer.2.normalization.weight: -0.105456
resnet.encoder.stages.1.layers.2.layer.2.normalization.bias: -0.780595
resnet.encoder.stages.1.layers.3.layer.0.convolution.weight: -0.015361
resnet.encoder.stages.1.layers.3.layer.0.normalization.weight: 1.799623
resnet.encoder.stages.1.layers.3.layer.0.normalization.bias: -1.113822
resnet.encoder.stages.1.layers.3.layer.1.convolution.weight: -0.008696
resnet.encoder.stages.1.layers.3.layer.1.normalization.weight: 1.754680
resnet.encoder.stages.1.layers.3.layer.1.normalization.bias: -0.441416
resnet.encoder.stages.1.layers.3.layer.2.convolution.weight: -0.000239
```

```
resnet.encoder.stages.1.layers.3.layer.2.normalization.weight: -0.008311
resnet.encoder.stages.1.layers.3.layer.2.normalization.bias: -0.980416
resnet.encoder.stages.2.layers.0.shortcut.convolution.weight: -0.008264
resnet.encoder.stages.2.layers.0.shortcut.normalization.weight: 1.764272
resnet.encoder.stages.2.layers.0.shortcut.normalization.bias: 0.187683
resnet.encoder.stages.2.layers.0.layer.0.convolution.weight: -0.011996
resnet.encoder.stages.2.layers.0.layer.0.normalization.weight: 2.024307
resnet.encoder.stages.2.layers.0.layer.0.normalization.bias: -1.499414
resnet.encoder.stages.2.layers.0.layer.1.convolution.weight: -0.002505
resnet.encoder.stages.2.layers.0.layer.1.normalization.weight: 2.020962
resnet.encoder.stages.2.layers.0.layer.1.normalization.bias: 0.905380
resnet.encoder.stages.2.layers.0.layer.2.convolution.weight: -0.000425
resnet.encoder.stages.2.layers.0.layer.2.normalization.weight: 0.061597
resnet.encoder.stages.2.layers.0.layer.2.normalization.bias: -1.657198
resnet.encoder.stages.2.layers.1.layer.0.convolution.weight: -0.011046
resnet.encoder.stages.2.layers.1.layer.0.normalization.weight: 1.947602
resnet.encoder.stages.2.layers.1.layer.0.normalization.bias: -0.728251
resnet.encoder.stages.2.layers.1.layer.1.convolution.weight: -0.008191
resnet.encoder.stages.2.layers.1.layer.1.normalization.weight: 1.985612
resnet.encoder.stages.2.layers.1.layer.1.normalization.bias: -0.807830
resnet.encoder.stages.2.layers.1.layer.2.convolution.weight: -0.000150
resnet.encoder.stages.2.layers.1.layer.2.normalization.weight: -0.033767
resnet.encoder.stages.2.layers.1.layer.2.normalization.bias: -1.123673
resnet.encoder.stages.2.layers.2.layer.0.convolution.weight: -0.012246
resnet.encoder.stages.2.layers.2.layer.0.normalization.weight: 1.973454
resnet.encoder.stages.2.layers.2.layer.0.normalization.bias: -1.203960
resnet.encoder.stages.2.layers.2.layer.1.convolution.weight: -0.009464
resnet.encoder.stages.2.layers.2.layer.1.normalization.weight: 1.634962
resnet.encoder.stages.2.layers.2.layer.1.normalization.bias: -0.933034
resnet.encoder.stages.2.layers.2.layer.2.convolution.weight: 0.000881
resnet.encoder.stages.2.layers.2.layer.2.normalization.weight: -0.031099
resnet.encoder.stages.2.layers.2.layer.2.normalization.bias: -0.945790
resnet.encoder.stages.2.layers.3.layer.0.convolution.weight: -0.012818
resnet.encoder.stages.2.layers.3.layer.0.normalization.weight: 1.896807
resnet.encoder.stages.2.layers.3.layer.0.normalization.bias: -1.462576
resnet.encoder.stages.2.layers.3.layer.1.convolution.weight: -0.009576
resnet.encoder.stages.2.layers.3.layer.1.normalization.weight: 1.810688
```

```
resnet.encoder.stages.2.layers.3.layer.1.normalization.bias: -0.800528
resnet.encoder.stages.2.layers.3.layer.2.convolution.weight: -0.000116
resnet.encoder.stages.2.layers.3.layer.2.normalization.weight: -0.055981
resnet.encoder.stages.2.layers.3.layer.2.normalization.bias: -1.390878
resnet.encoder.stages.2.layers.4.layer.0.convolution.weight: -0.011292
resnet.encoder.stages.2.layers.4.layer.0.normalization.weight: 1.904061
resnet.encoder.stages.2.layers.4.layer.0.normalization.bias: -1.611384
resnet.encoder.stages.2.layers.4.layer.1.convolution.weight: -0.009714
resnet.encoder.stages.2.layers.4.layer.1.normalization.weight: 1.933297
resnet.encoder.stages.2.layers.4.layer.1.normalization.bias: -0.423878
resnet.encoder.stages.2.layers.4.layer.2.convolution.weight: -0.000745
resnet.encoder.stages.2.layers.4.layer.2.normalization.weight: 0.009311
resnet.encoder.stages.2.layers.4.layer.2.normalization.bias: -1.628594
resnet.encoder.stages.2.layers.5.layer.0.convolution.weight: -0.013745
resnet.encoder.stages.2.layers.5.layer.0.normalization.weight: 1.785930
resnet.encoder.stages.2.layers.5.layer.0.normalization.bias: -1.842249
resnet.encoder.stages.2.layers.5.layer.1.convolution.weight: -0.008862
resnet.encoder.stages.2.layers.5.layer.1.normalization.weight: 1.909695
resnet.encoder.stages.2.layers.5.layer.1.normalization.bias: -0.283418
resnet.encoder.stages.2.layers.5.layer.2.convolution.weight: -0.000261
resnet.encoder.stages.2.layers.5.layer.2.normalization.weight: 0.031236
resnet.encoder.stages.2.layers.5.layer.2.normalization.bias: -1.398134
resnet.encoder.stages.3.layers.0.shortcut.convolution.weight: -0.018372
resnet.encoder.stages.3.layers.0.shortcut.normalization.weight: 1.529987
resnet.encoder.stages.3.layers.0.shortcut.normalization.bias: -1.791697
resnet.encoder.stages.3.layers.0.layer.0.convolution.weight: -0.013308
resnet.encoder.stages.3.layers.0.layer.0.normalization.weight: 1.770344
resnet.encoder.stages.3.layers.0.layer.0.normalization.bias: -2.061499
resnet.encoder.stages.3.layers.0.layer.1.convolution.weight: -0.010360
resnet.encoder.stages.3.layers.0.layer.1.normalization.weight: 2.157347
resnet.encoder.stages.3.layers.0.layer.1.normalization.bias: -0.604270
resnet.encoder.stages.3.layers.0.layer.2.convolution.weight: -0.000852
resnet.encoder.stages.3.layers.0.layer.2.normalization.weight: 0.041387
resnet.encoder.stages.3.layers.0.layer.2.normalization.bias: -1.972138
resnet.encoder.stages.3.layers.1.layer.0.convolution.weight: -0.021782
resnet.encoder.stages.3.layers.1.layer.0.normalization.weight: 2.100235
resnet.encoder.stages.3.layers.1.layer.0.normalization.bias: -1.166228
```

```
resnet.encoder.stages.3.layers.1.layer.1.convolution.weight: -0.016316
resnet.encoder.stages.3.layers.1.layer.1.normalization.weight: 1.772118
resnet.encoder.stages.3.layers.1.layer.1.normalization.bias: -1.445984
resnet.encoder.stages.3.layers.1.layer.2.convolution.weight: 0.000836
resnet.encoder.stages.3.layers.1.layer.2.normalization.weight: -0.043169
resnet.encoder.stages.3.layers.1.layer.2.normalization.bias: -1.786885
resnet.encoder.stages.3.layers.2.layer.0.convolution.weight: -0.029451
resnet.encoder.stages.3.layers.2.layer.0.normalization.weight: 1.824943
resnet.encoder.stages.3.layers.2.layer.0.normalization.bias: -1.627092
resnet.encoder.stages.3.layers.2.layer.1.convolution.weight: -0.018050
resnet.encoder.stages.3.layers.2.layer.1.normalization.weight: 1.655560
resnet.encoder.stages.3.layers.2.layer.1.normalization.bias: -1.362317
resnet.encoder.stages.3.layers.2.layer.2.convolution.weight: -0.000676
resnet.encoder.stages.3.layers.2.layer.2.normalization.weight: 0.003685
resnet.encoder.stages.3.layers.2.layer.2.normalization.bias: -1.391207
classifier.1.weight: -0.000336
classifier.1.bias: -0.001342
```

TRAINING CODE :

```python
# Training function
def train_model(model, train_loader, epochs=5, lr=1e-4):
    model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=lr)

    model.train()
    for epoch in range(epochs):
        running_loss = 0.0
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = model(images).logits
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()

        print(f"Epoch {epoch+1}, Loss: {running_loss/len(train_loader)}")

    return model
```

Train model(created in the above point) on a downloaded dataset, without augmentation.

```
Training on dataset without augmentation...
Epoch 1, Loss: 0.6802100794655936
Epoch 2, Loss: 0.6520122374807086
Epoch 3, Loss: 0.6242221934454781
```

Train model(created in the first point) on a downloaded dataset with augmentation.

```
Training on dataset with augmentation...
Epoch 1, Loss: 0.613908344791049
Epoch 2, Loss: 0.5475856023175376
Epoch 3, Loss: 0.45314411180359976
```

EVALUATION

Get the precision, recall, F1 score, and accuracy of both the models on the test set.

```
Performance on Test Set
--------------------------------------------------
Model Trained Without Augmentation:
Accuracy: 0.8571, Precision: 0.8646, Recall: 0.8571, F1 Score: 0.8564

Model Trained With Augmentation:
Accuracy: 0.9286, Precision: 0.9375, Recall: 0.9286, F1 Score: 0.9282
```

The model trained on augmented data performed better because augmentation introduced greater data diversity, making it more robust to real-world variations. By exposing the model to transformations like rotation, cropping, and brightness changes, it learned to focus on essential object features rather than memorizing specific patterns, thereby reducing overfitting. This improved generalization enabled the model to recognize test images more accurately, even if they had slight differences from the training set. Additionally, augmentation ensured that minor distortions did not mislead the model, leading to stronger representation learning. As a result, the model adapted better to unseen data, ultimately improving its accuracy and overall performance.
