IMAGE DATA AUGMENTATION AND IMAGE CLASSIFICATION

Shreya Sathapathi

B.Tech Computer Science and Engineering,
SRM Institute of Science and Technology, Ramapuram Part-Vadapalani

Period of Internship: 21ST January 2026 – 17TH February 2026

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata

1. ABSTRACT

This project focuses on image preprocessing, data augmentation, and dataset preparation for image classification tasks. OpenCV was used to perform basic image processing operations such as grayscale conversion, resizing, shifting, and rotation. A Cat and Dog image dataset was downloaded and prepared for classification using PyTorch and Torchvision. Data augmentation techniques such as resizing and random horizontal flipping were implemented to improve model generalization. The ImageFolder class was used to automatically assign labels based on directory structure. A DataLoader was created to generate batches of images for training. The project demonstrates foundational concepts required for deep learning-based image classification systems. The workflow highlights preprocessing steps necessary before training convolutional neural networks. This project provides hands-on experience in handling real-world image datasets using Python libraries.

2. INTRODUCTION

Image classification is one of the most important applications of computer vision and deep learning. Before training a classification model, images must undergo preprocessing and augmentation to ensure consistency and improve performance. This project demonstrates the process of image manipulation using OpenCV and dataset preparation using PyTorch. Various image transformation techniques such as grayscale conversion, shifting, resizing, and rotation were implemented. These transformations help in understanding geometric operations applied to digital images. The Cat and Dog dataset was used to demonstrate dataset handling and augmentation techniques. Torchvision's transformation pipeline was used to resize images, apply random horizontal flipping, and convert images into tensors. The ImageFolder class automatically assigns labels based on folder names, making it convenient for classification tasks.

Training Topics Covered During First Two Weeks:

- **Python Basics:** Variables, data types, control structures, functions, and basic programming logic.
- **Introduction to Machine Learning**: Basic concepts of supervised and unsupervised learning, model evaluation, and real-world applications.
- **Introduction to Large Language Models (LLMs)**: Overview of generative AI, transformer models, and applications of LLMs in industry.
- **Communication Skills:** Effective presentation techniques, technical explanation methods, and professional communication practices.

3. PROJECT OBJECTIVE

The objectives of this project are:

- To understand image loading and preprocessing using OpenCV.
- To perform grayscale conversion and geometric transformations.
- To implement image shifting, resizing, and rotation.
- To download and structure a real-world image dataset.
- To apply data augmentation techniques using Torchvision.
- To create an ImageFolder dataset for automatic label assignment.
- To generate batches of images using PyTorch DataLoader.

4. METHODOLOGY

The project was implemented using Python in Google Colab.

The methodology followed is described below:

- Step 1: Image Loading: The image 'moon-pexels-frank-cone.jpg' was loaded using OpenCV's cv2.imread() function. The shape of the image was printed to verify dimensions.
- Step 2: Grayscale Conversion: The image was converted into grayscale using cv2.cvtColor(). This reduces the image from three channels (RGB) to a single intensity channel.
- Step 3: Saving Processed Image: The grayscale image was saved using cv2.imwrite().
- Step 4: Image Shifting: A transformation matrix was created to shift the image 50 pixels right and 100 pixels down using cv2.warpAffine().
- Step 5: Image Resizing: The image was resized to 150x100 pixels using cv2.resize().
- Step 6: Image Rotation: The image was rotated 90 degrees counter-clockwise using cv2.getRotationMatrix2D() and cv2.warpAffine().
- Step 7: Dataset Download: The Cat and Dog dataset was downloaded using wget and extracted using unzip in Google Colab.
- Step 8: Data Augmentation: A transformation pipeline was created using Torchvision transforms including: Resize to 255x255 Random horizontal flip Conversion to tensor.
- Step 9: Dataset Creation: The ImageFolder class was used to create a training dataset from the extracted folder structure.

- Step 10: DataLoader Creation: A DataLoader was created with batch size 64 and shuffle enabled to generate image batches for training. Tools Used: Python, NumPy, OpenCV, PyTorch, Torchvision, Google Colab.

5. DATA ANALYSIS AND RESULTS

Descriptive Analysis:

- The original moon image had three channels representing RGB.
- After grayscale conversion, the image contained only two dimensions (height and width).
- Resizing successfully changed the image dimensions to (100, 150, 3).
- Rotation swapped the width and height dimensions.

Dataset Analysis:

- The Cat and Dog dataset was successfully downloaded and extracted.
- The ImageFolder dataset correctly identified image classes based on folder names.
- The total number of images in the training dataset was verified using len(train_dataset).

Batch Output Verification:

The DataLoader generated batches of shape:

Images: torch.Size([64, 3, 255, 255])

Labels: torch.Size([64])

This confirms: 64 images per batch 3 color channels 255x255 image dimensions.

These outputs validate correct preprocessing and batching of images for deep learning tasks.

5.1 Image Processing Results

The following image processing operations were performed using OpenCV:

(a) Grayscale Conversion

Figure 1: Grayscale Image Output

(b) Resized Image

Figure 2: Resized Image (150x100 pixels)

(c) Rotated Image

Figure 3: Rotated Image (90° Counter-Clockwise)

5.2 Dataset Preparation Results

(a) Dataset Size Verification

```
from torchvision import datasets

data_dir = 'Cat_Dog_data/train'

train_dataset = datasets.ImageFolder(data_dir, transform=train_transform)

print(len(train_dataset))
```

```
22500
```

Figure 4: Total Images in Training Dataset

(b) DataLoader Batch Output

```
from torch.utils.data import DataLoader

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)

# Gets one batch
images, labels = next(iter(train_loader))

print(images.shape)
print(labels.shape)
```

```
torch.Size([64, 3, 255, 255])
torch.Size([64])
```

Figure 5: DataLoader Batch Output

6. CONCLUSION

The project successfully demonstrates image preprocessing and dataset preparation for image classification tasks. Various image transformations such as grayscale conversion, shifting, resizing, and rotation were implemented using OpenCV. Data augmentation techniques were applied using Torchvision to improve dataset variability. The ImageFolder and DataLoader utilities simplified dataset handling and batch generation. This project provides foundational knowledge required before building deep learning models such as Convolutional Neural Networks. Future work can include training a CNN model on the prepared dataset and evaluating classification accuracy.

7. APPENDICES

Appendix 1 – References

- OpenCV Documentation – https://docs.opencv.org
- PyTorch Documentation – https://pytorch.org/docs
- Torchvision Documentation – https://pytorch.org/vision/stable

Appendix 2 – GitHub Link

- GitHub Repository: https://github.com/ShreyaSathapathi06/Spring-Internship-2026-Image-Classification