Addis Ababa University

Master's in Artificial Intelligence

Computer Vision (CV) Laboratory Manual

Prepared by: Dr. Natnael Argaw Wondimu

Environment: OpenCV and Visual Studio

Preface

This manual is prepared in a workbook style for AI Master's students at Addis Ababa University. It provides a hands-on introduction to Computer vision with OpenCV, covering advanced topics. Each lab includes objectives, theoretical background, procedures, OpenCV code, checkpoints, try-it-yourself prompts, and collaborative assignments. The manual is intended to serve as a base for advanced studies in Computer Vision.

Chapter 10: Introduction to Deep Learning for Vision (PyTorch / TensorFlow)

Objective

To introduce the fundamentals of deep learning in computer vision using Convolutional Neural Networks (CNNs) and pretrained models. Students will learn image classification workflows using PyTorch or TensorFlow and apply them to real-world images.

1. What is Deep Learning for Vision?

Description: Deep learning enables automated feature extraction from images using multi-layer neural networks, especially **Convolutional Neural Networks (CNNs)**, which are designed to handle spatial data like images.

2. Environment Setup

2.1 Required Libraries

```
pip install torch torchvision torchaudio matplotlib
pip install tensorflow keras opencv-python
```

Choose either **PyTorch** or **TensorFlow** as your preferred framework.

3. CNN-Based Image Classification with PyTorch

3.1 Using Pretrained ResNet for Inference

```
import torch
import torchvision.transforms as transforms
from PIL import Image
from torchvision import models
import matplotlib.pyplot as plt

# Load pretrained model
model = models.resnet18(pretrained=True)
model.eval()

# Image preprocessing
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

img = Image.open('dog.jpg')
img_tensor = transform(img).unsqueeze(0)

# Predict
with torch.no_grad():
    output = model(img_tensor)
    _, predicted = output.max(1)
```

```
print('Predicted class index:', predicted.item())
```

Output Description: Classifies the input image using ResNet-18 and outputs the predicted class index.

4. CNN-Based Image Classification with TensorFlow/Keras

4.1 Using Pretrained MobileNetV2

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image
import numpy as np

model = MobileNetV2(weights='imagenet')
img = image.load_img('cat.jpg', target_size=(224, 224))

x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

Output Description: Displays the top-3 predicted labels and probabilities using MobileNetV2.

5. Custom Image Input with OpenCV

```
import cv2
img = cv2.imread('test.jpg')
img = cv2.resize(img, (224, 224))
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# Continue with same preprocessing and prediction as above
```

Output Description: Enables using OpenCV images as inputs for CNNs in both frameworks.

6. Visualizing CNN Predictions

```
plt.imshow(img)
plt.title(f"Predicted: {decode_predictions(preds, top=1)[0][0][1]}")
plt.axis('off')
plt.show()
```

Output Description: Displays the input image with the top predicted class label.

7. Summary

- **CNNs** automate feature extraction using layers of convolutions and activations.
- **Pretrained models** like ResNet and MobileNet reduce training time and improve accuracy.
- PyTorch and TensorFlow provide easy access to top-performing models.

Suggested Exercises

1. Run predictions on your own image dataset.
2. Compare predictions from ResNet and MobileNet.
3. Fine-tune a pretrained model on a small custom dataset.
4. Visualize activation maps or intermediate feature layers.