NAME: KSHITIJ VINOD SALI
CLASS: BE-A ROLL NO.:
SUBJECT: LP-IV: DEEP LEARNING

LAB ASSIGNMENT - 06

Title: Object Detection using CNN-based Transfer Learning.

Course Outcome: CO2 - Build and train a deep neural network models for use in various applications
CO3 - Apply deep learning techniques like CNN, RNN auto encoders to solve real world problems.

Date of Completion:
Assessment Grade/ Marks:
Assessor's Sign with Date:

Problem Statement: Object detection using Transfer Learning of CNN architectures.

a) Load in pre-trained CNN model trained on a large dataset.
b) Freeze parameters (weights) in model's lower convolutional layers.
c) Add custom classifier with several layers of trainable parameters to model.
d) Train classifier layers on training data available for task.
e) Fine-tune hyper parameters & unfreeze more layers as needed.

Blooms Taxanomy Category: Create, Apply.

Requirements: Python libraries, dataset, pre-trained model.

Theory:

i) What is Transfer learning?

→ Transfer learning is a machine learning techniques where a model developed for one task is reused as starting point for a model on a second, related task. Instead of building a model from scratch, you leverage "knowledge" learned from a large dataset to improve the performance on new task, which often has less data.

ii) What are pretrained Neural Network models?

→ A pretrained model is a neural network that has already be trained on very large, general-purpose dataset. These model have already learned to detect universal features like edges, texture & patterns.

iii) Explain Pytorch in short.

→ Pytorch is a popular open-source machine learning library for Python. It is known for its flexibility and ease of use, particularly in research. Its key features include:

a) Tensor , b) Dynamic computation graph.
c) autograd.

iv) What are advantages of Transfer learning?
→ a) Reduced training time.
b) Less data required.
c) Better performance.
d) Faster development.

v) What are applications of Transfer learning?
→ a) Computer vision
b) Natural Language Processing (NLP)
c) Audio processing.

vi) Explain Caltech 101 image dataset.
→ The Caltech 101 dataset is a classic computer vision dataset containing pictures of objects belonging to 101 different categories, plus one background category. It has roughly 40 to 800 images per category, totally around 9,000 images. It was primarily created for tasks like object recognition & classification.

vii) Explain ImageNet dataset.
→ ImageNet is a massive dataset of over 14 million image, hand-annotated to indicate what object are pictured. The image are organized into more than 20,000 categories. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) uses a subset of ImageNet & has been instrumental in benchmarking the performance of computer vision models. Most state-of-the-art pretrained models are trained on this dataset.

viii) List down basic steps of Transfer learning.

→ a) Select a pretrained model.
b) Freeze weights of early layers.
c) Replace the classifier.
d) Train the new layers.
e) Unfreeze some of later layers of original model.

ix) What is data augumentation?

→ Data augumentation is a technique used to artificially increase the size of diversity of training dataset. It works by creating slightly modified copies of existing data.

x) How and why data augumentation is done related to transfer learning?

→ Why - To prevent overfitting. When your dataset is small, the model might just memorize the training images.
How - By applying random, minor transformations like filtering, rotating or changing colours to your training images.

xi) Why preprocessing is needed on input data in Transfer learning?

→ Preprocessing is crucial because a pretrained model expects input data to be in exact same format as image (data) it was originally trained on. This includes image size and normalization. If the new data is not prepro-cced correctly, the pretrained feature

extraction will not work as intended, leading to poor performance.

xii) What is PyTorch Transforms module. Explain following commands w.r.t. it:
compose ([RandomResizeCrop (size = 256, scale = (0.8, 1.0)), RandomRotation (degrees = 15),
colorJitter (),
randomHorizontalFlip (),
centreCrop (size = 224), # Image net standards
.toTensor (),
normalize

→ The torchvision.transform module provides tools to preprocess & augumented images for your model. The compose function chains these transformations together into a pipeline.

Commands -

- compose ([...]) - Combines multiple image transformations to apply them in sequence.
- randomResized Crop (256) - Randomly crops part of image & resizes it to 256 x 256 pixels.
- randomRotation (15) - Rotates image randomly between -15° to +15°
- colorJitter () - Randomly adjust brightness, contrast & saturation.
- randomHorizontalFlip () - Flips images horizontally with 50% chance.
- centerCrop (224) - Takes a centred 224 x 224 crop from image.
- toTensor () - Converts image to PyTorch tensor &

scales pixel values to [0,1].
- Normalizer(...) - Standardize pixel values using mean & standard deviation for each color channel

xiii) Explain the validation transformation steps with PyTorch transforms.
→ For validation or testing, we avoid random transformation to ensure consistent results. The typical steps are:
a) Resize (256 x 256) - Resizes image to fixed size.
b) CentreCrop (224 x 224) - Crops centre region to match model input.
c) ToTenser() - Converts image to PyTorch tensor.
d) Normalizer(...) - Standardizes pixel values using mean & standard deviation as training.

xiv) Explain VGG-16 model from PyTorch.
→ VGG-16 is a deep convolutional neural network (CNN) architecture developed by the Visual Geometry Group at Oxford. The "16" refers to its 16 layers with trainable weights (13 convolutional layers & 3 fully-connected layers).

It uses simple & effective design makes it widely used for image classification & transfer learning.

It uses small 3x3 filters, stacked to capture complex features efficiently.

Algorithm and Steps:

i) Prepare the dataset in splitting in three directories: Train, Alidation and Test.
ii) Do prerpocessing on data with transform from PyTorch Training dataset transformation.
iii) Create datasets & loaders.
iv) Load pretrained model.
v) Freeze all the model weights.
vi) Add custom classifier with parameters.
vii) Only train the first six layer of classifier & keep remaining layer off.
viii) Initialize the loss & optimizer criteration
ix) Train the model using PyTorch for epochs
x) Perform early stopping.
xi) Draw performance curve.
xii) Calculate accuracy.

Inference: In this experiment, we were able to see basics of using PyTorch as well as concept of transfer learning, an effective method for object recognition. Instead of training a model from scratch, we can use existing architectures that have been trained on large dataset & then tune them for our task. This reduces time to train & often results in better overall performance. The outcome of this experiment is knowledge to transfer learning and PyTorch that we can build on to build more complex applications.