Lab Experiment 3

Title: Building a Cognitive-based Applicatior
Images

1. Aim

To design and implement a Cognitive Image-bas
that acquires knowledge from images for domai
Insurance, Healthcare, Smarter Cities, and Gc

2. Theory

Cognitive Computing refers to systems that ca
in analyzing data.

Image-based Cognitive Applications use Deep L
VGG, YOLO) to extract knowledge from visual c

Applications in different domains:

Healthcare → Detecting diseases from X-rays/M

Insurance → Assessing car damage for claims.

Customer Service → Reading scanned documents

Smarter Cities → Traffic monitoring with CCTV

Government → Satellite image classification f

Thus, cognitive models act as knowledge acqui
complex images like humans.

3. Requirements

Google Colab

Python Libraries: torch, torchvision, pillow,

4. Procedure (Google Colab Commands)
🔹 Step 1: Install Required Libraries

```
!pip install torch torchvision matplotlib pil
```

🔹 Step 2: Import Libraries

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

🔹 Step 3: Load Pre-trained Cognitive Model

Lab Experiment 3

Title: Building a Cognitive-based Application to Acquire Knowledge through Images

1. Aim

To design and implement a Cognitive Image-based Application using Deep Learning that acquires knowledge from images for domains such as Customer Service, Insurance, Healthcare, Smarter Cities, and Government, in Google Colab.

2. Theory

Cognitive Computing refers to systems that can simulate human thought processes in analyzing data.

Image-based Cognitive Applications use Deep Learning models (like CNNs: ResNet, VGG, YOLO) to extract knowledge from visual data.

Applications in different domains:

Healthcare → Detecting diseases from X-rays/MRIs.

Insurance → Assessing car damage for claims.

Customer Service → Reading scanned documents via OCR.

Smarter Cities → Traffic monitoring with CCTV.

Government → Satellite image classification for disaster management.

Thus, cognitive models act as knowledge acquisition systems by interpreting complex images like humans.

3. Requirements

Google Colab

```
 ◆ Step 3: Load Pre-trained Cognitive Model
# Load ResNet18 (pretrained on ImageNet)
model = models.resnet18(pretrained=True)
model.eval()

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

 ◆ Step 4: Upload an Image
from google.colab import files

# Upload an image (X-ray, car accident photo,
uploaded = files.upload()

# Open uploaded image
img_path = list(uploaded.keys())[0]
img = Image.open(img_path).convert("RGB")

plt.imshow(img)
plt.axis("off")
plt.show()

# Preprocess
input_tensor = transform(img).unsqueeze(0)

 ◆ Step 5: Run Prediction
# Run inference
output = model(input_tensor)
_, predicted = torch.max(output, 1)

# Load labels
!wget -q https://raw.githubusercontent.com/py
txt -O imagenet_classes.txt
with open("imagenet_classes.txt") as f:
    labels = [line.strip() for line in f.reac

print("Predicted Class:", labels[predicted.it

 ◆ Step 6: Extend for Specific Domains

(a) Customer Service / Insurance (OCR on Form

import easyocr
reader = easyocr.Reader(['en'])
result = reader.readtext(img_path)
print("Extracted Text:", result)
```

Python Libraries: torch, torchvision, pillow, matplotlib, easyocr

4. Procedure (Google Colab Commands)

◆ Step 1: Install Required Libraries !pip install torch torchvision matplotlib pillow easyocr

◆ Step 2: Import Libraries import torch import torchvision.models as models import torchvision.transforms as transforms from PIL import Image import matplotlib.pyplot as plt

◆ Step 3: Load Pre-trained Cognitive Model

Load ResNet18 (pretrained on ImageNet)

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

Image Preprocessing Pipeline

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

◆ Step 4: Upload an Image from google.colab import files

Upload an image (X-ray, car accident photo, form, traffic photo, etc.)

uploaded = files.upload()

Open uploaded image

```
(b) Smarter Cities (Object Detection with YOL

!git clone https://github.com/ultralytics/yol
%cd yolov5
!pip install -r requirements.txt
!python detect.py --weights yolov5s.pt --sour

5. Result

The model successfully processed the uploadec

It predicted the class of the image (example:
"traffic light").

With OCR, the system extracted text from uplc

With YOLO, the system detected objects in cit

6. Conclusion

The experiment demonstrates that a Cognitive-
effectively acquire knowledge from images acr

In Healthcare, it can detect diseases.

In Insurance, it can analyze car damage and r

In Customer Service, it can process documents

In Smarter Cities, it can detect vehicles anc

In Government, it can analyze satellite image

Thus, such applications mimic human cognitive
decision-making support.
```

img_path = list(uploaded.keys())[0] img = Image.open(img_path).convert("RGB")

plt.imshow(img) plt.axis("off") plt.show()

Preprocess

input_tensor = transform(img).unsqueeze(0)

- 🔹 Step 5: Run Prediction

Run inference

output = model(input_tensor) _, predicted = torch.max(output, 1)

Load labels

!wget -q https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt -O imagenet_classes.txt with open("imagenet_classes.txt") as f: labels = [line.strip() for line in f.readlines()]

print("Predicted Class:", labels[predicted.item()])

- 🔹 Step 6: Extend for Specific Domains

(a) Customer Service / Insurance (OCR on Forms):

import easyocr reader = easyocr.Reader(['en']) result = reader.readtext(img_path) print("Extracted Text:", result)

(b) Smarter Cities (Object Detection with YOLO):

!git clone https://github.com/ultralytics/yolov5 %cd yolov5 !pip install -r requirements.txt !python detect.py --weights yolov5s.pt --source ../{img_path}

5. Result

The model successfully processed the uploaded image.

It predicted the class of the image (example: "car bumper", "X-ray, chest", "traffic light").

With OCR, the system extracted text from uploaded documents.

With YOLO, the system detected objects in city traffic images.

6. Conclusion

The experiment demonstrates that a Cognitive-based Image Application can effectively acquire knowledge from images across multiple domains.

In Healthcare, it can detect diseases.

In Insurance, it can analyze car damage and read forms.

In Customer Service, it can process documents via OCR.

In Smarter Cities, it can detect vehicles and traffic congestion.

In Government, it can analyze satellite images.

Thus, such applications mimic human cognitive abilities and provide intelligent decision-making support.

```
!pip install torch torchvision matplotlib pillow easyocr
```

```
  Downloading easyocr-1.7.2-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/py
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/pyt
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/pyt
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/pyth
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.1
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/pyt
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/pytho
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/di
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.12/
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: scikit-image in /usr/local/lib/python3.12/dist-packa
Collecting python-bidi (from easyocr)
  Downloading python_bidi-0.6.6-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86
Requirement already satisfied: PyYAML in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: Shapely in /usr/local/lib/python3.12/dist-packages (
Collecting pyclipper (from easyocr)
  Downloading pyclipper-1.3.0.post6-cp312-cp312-manylinux_2_17_x86_64.manylinux2014
Collecting ninja (from easyocr)
  Downloading ninja-1.13.0-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python3.12/
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.12/dis
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.12/dist-p
Downloading easyocr-1.7.2-py3-none-any.whl (2.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 27.8 MB/s eta 0:00:00
Downloading ninja-1.13.0-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (1
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 180.7/180.7 kB 10.5 MB/s eta 0:00:00
Downloading pyclipper-1.3.0.post6-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 963.8/963.8 kB 47.6 MB/s eta 0:00:00
Downloading python_bidi-0.6.6-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_6
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 292.1/292.1 kB 17.5 MB/s eta 0:00:00
Installing collected packages: python-bidi, pyclipper, ninja, easyocr
Successfully installed easyocr-1.7.2 ninja-1.13.0 pyclipper-1.3.0.post6 python-bidi
```

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

```python
# Load a ResNet18 pretrained on ImageNet
model = models.resnet18(pretrained=True)
model.eval()

# Define image preprocessing steps
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])
```

```
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.ca
100%|██████████| 44.7M/44.7M [00:00<00:00, 104MB/s]
```

```
from google.colab import files

# Upload image (e.g., X-ray, car damage, scanned form, satellite image)
uploaded = files.upload()

# Open uploaded file safely
img_path = list(uploaded.keys())[0]
img = Image.open(img_path).convert("RGB")

plt.imshow(img)
plt.axis("off")
plt.show()

# Preprocess
input_tensor = transform(img).unsqueeze(0)
```

Choose Files mopuntains.jpg

```
# Run through model
output = model(input_tensor)
_, predicted = torch.max(output, 1)

# Load labels
!wget -q https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt -O ima
with open("imagenet_classes.txt") as f:
    labels = [line.strip() for line in f.readlines()]

print("Predicted Class:", labels[predicted.item()])
```

```
Predicted Class: alp
```

```
import easyocr
reader = easyocr.Reader(['en'])
result = reader.readtext(img_path)
print("Extracted Text:", result)
```

```
WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CPU. Note:
WARNING:easyocr.easyocr:Downloading detection model, please wait. This may take sever
Progress: |██████████████████████████████████████████████████| 100.0% CompleteWARNING
Progress: |██████████████████████████████████████████████████| 100.0% Complete/usr/lo
  warnings.warn(warn_msg)
Extracted Text: [([[np.int32(621), np.int32(425)], [np.int32(723), np.int32(425)], [r
```

Start coding or generate with AI.