Practical – 1

Aim: **Study and analysis of Google Colab/Jupyter notebook/Tensor Flow for Deep Learning.**

Solution:

Deep learning relies on powerful platforms and frameworks for efficient model development, training, and validation. Google Colab, Jupyter Notebook, and TensorFlow are among the top tools leveraged by data scientists and AI practitioners. Each of these tools serves unique purposes and collectively enhances the deep learning workflow. Below is an analysis of their features and roles.

1. Google Colab:

Google Colab is a cloud-based platform by Google that provides an online environment for writing and executing Python code in a notebook interface. Its ease of use and access to high-performance hardware make it a favorite for machine learning and deep learning tasks.

Key Features:

- Online Access: Operates entirely on the cloud, removing the need for local setup. Accessible from any location with internet connectivity.
- Hardware Support: Offers free access to GPUs and TPUs, enabling faster training for computationally demanding models.
- Google Drive Integration: Simplifies saving, organizing, and sharing files directly through Google Drive.
- Collaboration-Friendly: Allows multiple users to work together in real time, similar to Google Docs.
- Preloaded Libraries: Includes popular deep learning libraries like TensorFlow, PyTorch, and Keras, reducing setup time.

Role in Deep Learning:

- Model Training: Optimized for training deep learning models using GPUs and TPUs.
- Quick Testing: Supports rapid experimentation without local hardware constraints.
- Integrated Notebooks: Provides a unified environment for coding, running, and visualizing results.

2. Jupyter Notebook:

An open-source tool, Jupyter Notebook provides an interactive platform for combining live code, markdown, visualizations, and other outputs in a single document. It is widely adopted for machine learning and data analysis tasks.

Key Features:

- Interactive Coding: Supports execution of code in small sections, enabling iterative testing and debugging.
- Rich Media Support: Facilitates inclusion of graphs, images, and other media, critical for data visualization and analysis.
- Flexible Documentation: Combines executable code with markdown for detailed, inline documentation.
- Broad Compatibility: Integrates seamlessly with various ML libraries, including TensorFlow and Matplotlib.

Role in Deep Learning:

- Data Analysis: An essential tool for exploratory data analysis (EDA) before model training.
- Iterative Development: Enables real-time testing and fine-tuning of model architectures and parameters.
- Result Visualization: Useful for presenting training curves, evaluation metrics, or insights from neural networks.

3. **TensorFlow:**

TensorFlow, developed by Google, is a versatile open-source library for building and deploying machine learning models. Its scalability and high-performance capabilities make it a cornerstone for deep learning projects.

Key Features:

- Comprehensive Toolkit: Provides tools for creating and training models, as well as APIs like Keras for high-level development.
- Scalability: Adapts to small-scale or distributed multi-GPU setups.
- Deployment Flexibility: Supports deployment through TensorFlow Serving, Lite, and.js for production, mobile, and web environments.
- Customizable: Offers both low-level and high-level control for novice and expert users alike.

Role in Deep Learning:

- Model Creation: Simplifies building and training models across various domains such as NLP and computer vision.
- Performance Optimization: Accelerates training through GPU/TPU support.
- Distributed Training: Efficiently processes large datasets using multiple hardware setups.
- Deployment Ready: Facilitates production-level deployment of models on diverse platforms.

Integration of Google Colab, Jupyter, and TensorFlow:

- Colab and Jupyter: Colab extends Jupyter's features into a cloud-based environment, making it ideal for TensorFlow applications.
- TensorFlow and Jupyter: Jupyter's interactive environment complements TensorFlow's model-building and visualization capabilities.
- Colab and TensorFlow: Offers an out-of-the-box TensorFlow setup, leveraging cloud hardware for efficient training and experimentation.

Advantages of Using These Tools:

- User-Friendly: Intuitive interfaces make them accessible to learners and professionals.
- Cloud Resources: Colab's free GPU/TPU support handles computation-heavy tasks.
- Collaborative Features: Simplifies team projects through shared access.
- Comprehensive Framework: TensorFlow's versatile features address the entire deep learning lifecycle.

Conclusion:

- Google Colab, Jupyter Notebook, and TensorFlow collectively form a powerful ecosystem for deep learning practitioners. Their seamless integration supports every step of the deep learning pipeline, from data exploration and model training to deployment.

Practical – 2

Aim: Import and Utilize the Data (CSV and Image) from Dataset Repository

Code:

Step – 1: Organizing the data-set

```
import os
import pandas as pd
dataset_path = 'C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101'

images_path = []
labels = []

for category in os.listdir(dataset_path):
    category_path = os.path.join(dataset_path, category)
    if os.path.isdir(category_path):
        for image_name in os.listdir(category_path):
            if image_name.endswith(".jpg"):
                images_path.append(os.path.join(category_path,image_name))
                labels. append (category)

data = pd.DataFrame({
    'image_path' : images_path,
    'labels' : labels
})

data.to_csv('data.csv', index = False)
```

Step – 2: Processing the images to certain size and setting up a path.

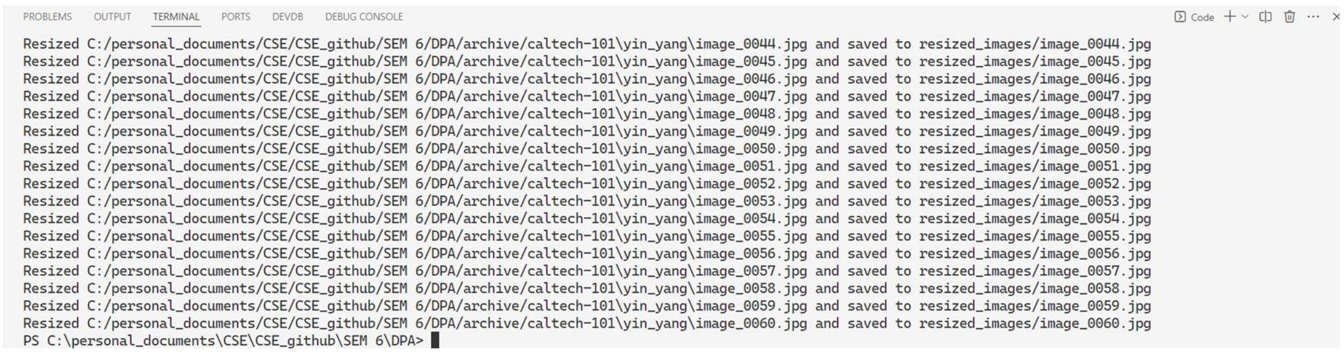
```
import os
from PIL import Image
from prac_2_gen_csv import data

desired_size = (128, 128)
def resize_image(image_path, output_path, desired_size):
    with Image.open(image_path) as img:
        resized_img = img.resize(desired_size)
        resized_img.save(output_path)

os.makedirs('resized_images', exist_ok = True)

for idx, row in data.iterrows():
    image_path = row['image_path']
    label = row['labels']
    output_path = f"resized_images/{os.path.basename(image_path)}"
    resize_image(image_path, output_path, desired_size)
    print (f"Resized {image_path} and saved to {output_path}")
```

Output:



```
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0044.jpg and saved to resized_images/image_0044.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0045.jpg and saved to resized_images/image_0045.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0046.jpg and saved to resized_images/image_0046.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0047.jpg and saved to resized_images/image_0047.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0048.jpg and saved to resized_images/image_0048.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0049.jpg and saved to resized_images/image_0049.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0050.jpg and saved to resized_images/image_0050.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0051.jpg and saved to resized_images/image_0051.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0052.jpg and saved to resized_images/image_0052.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0053.jpg and saved to resized_images/image_0053.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0054.jpg and saved to resized_images/image_0054.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0055.jpg and saved to resized_images/image_0055.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0056.jpg and saved to resized_images/image_0056.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0057.jpg and saved to resized_images/image_0057.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0058.jpg and saved to resized_images/image_0058.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0059.jpg and saved to resized_images/image_0059.jpg
Resized C:/personal_documents/CSE/CSE_github/SEM 6/DPA/archive/caltech-101\yin_yang\image_0060.jpg and saved to resized_images/image_0060.jpg
PS C:\personal_documents\CSE\CSE_github\SEM 6\DPA> 
```

Step – 3: Displaying random Images

```
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt
import random as rand

data = pd.read_csv("data.csv")

print(data.head())

last = data.iloc[-1]
random_number = rand.randint(0, last.name)
first_img_path = data.loc[random_number, 'image_path']
label = data.loc[random_number, 'labels']

image = Image.open(first_img_path)
plt.imshow(image)
plt.title(f"Label: {label}")
plt.axis('off')
plt.show()
```

Output:

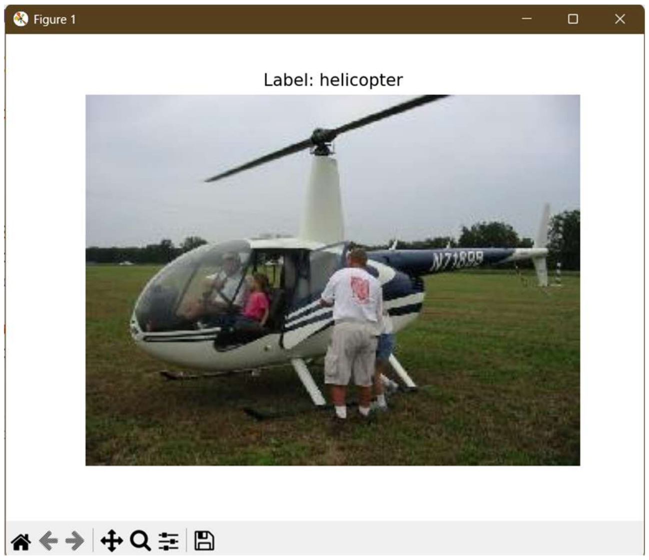


Figure 1
Label: helicopter
N7189B

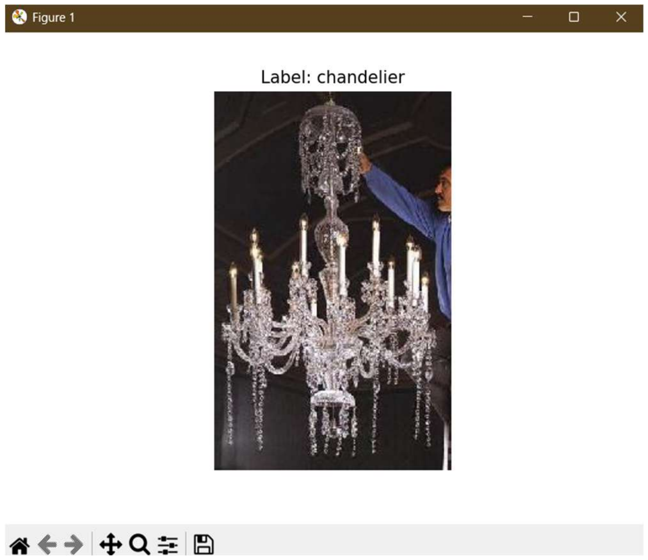


Figure 1
Label: chandelier