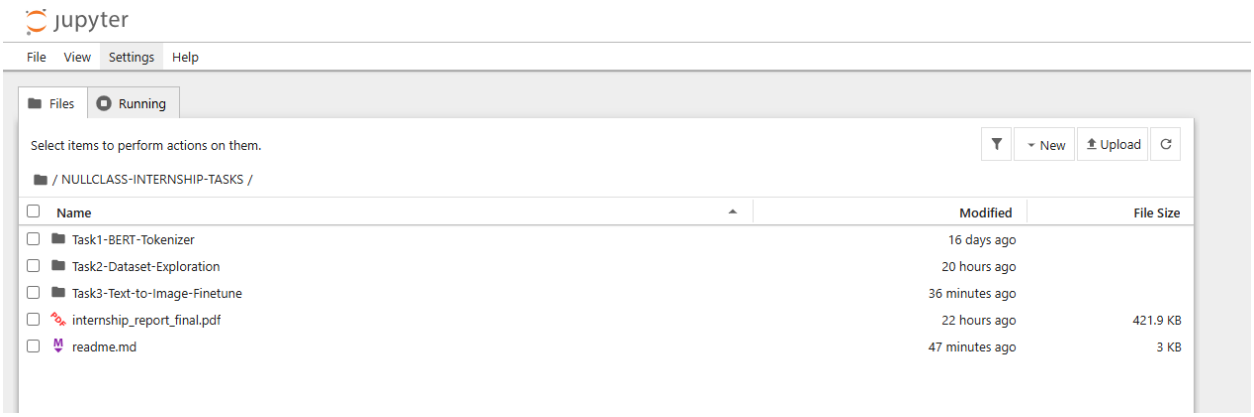
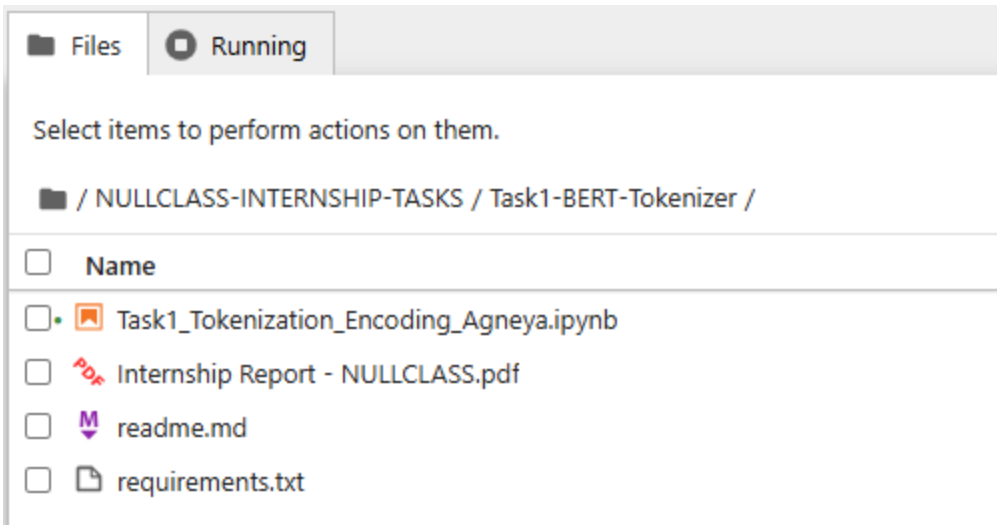
SCREENSHOTS

AUTHOR : AGNEYA S NAMBIAR






```
[1]: import sys
!{sys.executable} -m pip install transformers torch
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: transformers in c:\users\user\appdata\roaming\python\python312\site-packages (4.52.4)
Requirement already satisfied: torch in c:\users\user\appdata\roaming\python\python312\site-packages (2.7.0)
Requirement already satisfied: filelock in c:\programdata\anaconda3\lib\site-packages (from transformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in c:\users\user\appdata\roaming\python\python312\site-packages (from transformers) (0.32.
3)
Requirement already satisfied: numpy>=1.17 in c:\programdata\anaconda3\lib\site-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in c:\programdata\anaconda3\lib\site-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in c:\programdata\anaconda3\lib\site-packages (from transformers) (2024.9.11)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in c:\users\user\appdata\roaming\python\python312\site-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in c:\users\user\appdata\roaming\python\python312\site-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in c:\programdata\anaconda3\lib\site-packages (from transformers) (4.66.5)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\programdata\anaconda3\lib\site-packages (from torch) (4.11.0)
Requirement already satisfied: sympy>=1.13.3 in c:\users\user\appdata\roaming\python\python312\site-packages (from torch) (1.14.0)
Requirement already satisfied: networkx in c:\programdata\anaconda3\lib\site-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in c:\programdata\anaconda3\lib\site-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in c:\programdata\anaconda3\lib\site-packages (from torch) (2024.6.1)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from torch) (75.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests->transformers) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests->transformers) (2024.8.30)
```

```
[1]: !pip install huggingface_hub[hf_xet]
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: huggingface_hub[hf_xet] in c:\users\user\appdata\roaming\python\python312\site-packages (0.32.3)
Requirement already satisfied: filelock in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (3.13.1)
Requirement already satisfied: fsspec>=2023.5.0 in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (2024.6.1)
Requirement already satisfied: packaging>=20.9 in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (24.1)
Requirement already satisfied: pyyaml>=5.1 in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (6.0.1)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (4.66.5)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\programdata\anaconda3\lib\site-packages (from huggingface_hub[hf_xet]) (4.11.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in c:\users\user\appdata\roaming\python\python312\site-packages (from huggingface_hub[hf_xet]) (1.
1.2)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from tqdm>=4.42.1->huggingface_hub[hf_xet]) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests->huggingface_hub[hf_xet]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests->huggingface_hub[hf_xet]) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests->huggingface_hub[hf_xet]) (2.2.3)
```

[11]:
```
from transformers import BertTokenizer, BertModel
import torch
import numpy
import pandas
```

[12]:
```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
```

[13]:
```
text = "Hello, my name is agneya and i am doing internship with nullclass"
```

[14]:
```
#Step 1: Tokenization (splitting into tokens)
tokens = tokenizer.tokenize(text)
print("Tokens:")
print(tokens)
```

```
Tokens:
['hello', ',', 'my', 'name', 'is', 'ag', '##ney', '##a', 'and', 'i', 'am', 'doing', 'internship', 'with', 'null', '##class']
```

[15]:
```
#Step 2: Encoding (tokens → input IDs + tensors)
encoding = tokenizer(text, return_tensors='pt')
```

[16]:
```
print("Input IDs:")
print(encoding['input_ids'])
```

```
Input IDs:
tensor([[  101,  7592,  1010,  2026,  2171,  2003, 12943,  5420,  2050,  1998,
          1045,  2572,  2725, 22676,  2007, 19701, 26266,   102]])
```

[17]:
```
print("Attention Mask:")
print(encoding['attention_mask'])
```

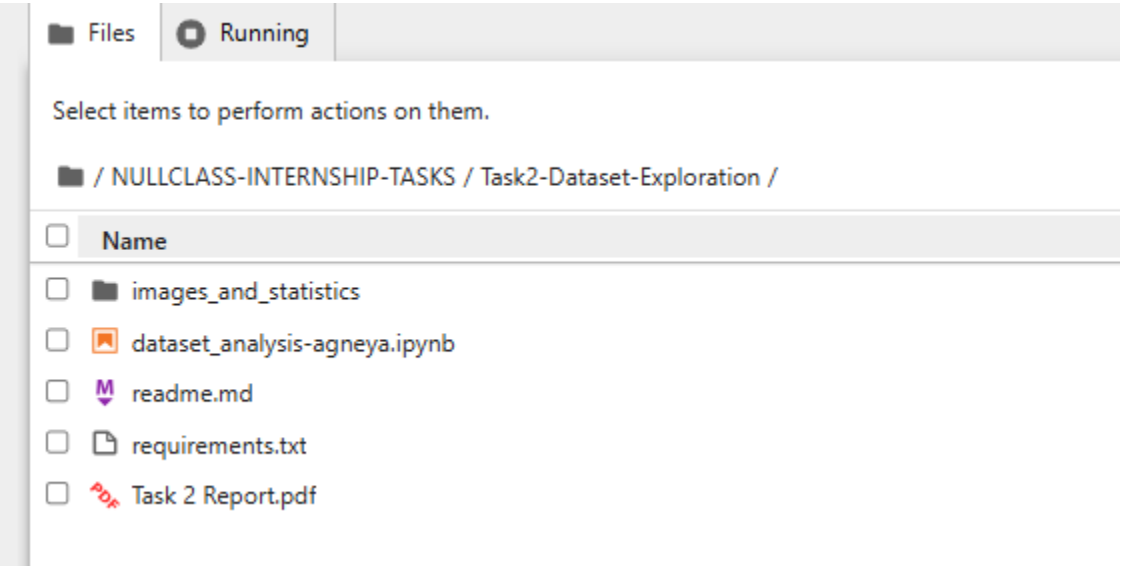
```
Attention Mask:
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
```

[18]:
```
#Step 3:Use BERT to get embeddings
with torch.no_grad():
    outputs = model(**encoding)
print("\nOutput Embeddings Shape (last_hidden_state):")
print(outputs.last_hidden_state.shape)
```

```
Output Embeddings Shape (last_hidden_state):
torch.Size([1, 18, 768])
```

[19]:
```
#tokenizer.tokenize()=Splits sentence into wordpieces
#tokenizer(text, return_tensors='pt')=Converts text into token IDs and attention mask
#model(**encoding)=Feeds input to BERT to get embeddings
#outputs.last_hidden_state.shape=Returns [1, sequence_length, 768] – the BERT embeddings
```

TASK 2



[2]:
```
pip install tensorflow-datasets tensorflow matplotlib pandas numpy
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow-datasets in c:\users\user\appdata\roaming\python\python312\site-packages (4.9.9)
Requirement already satisfied: tensorflow in c:\users\user\appdata\roaming\python\python312\site-packages (2.19.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (3.9.2)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (1.26.4)
Requirement already satisfied: absl-py in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-datasets) (2.3.0)
Requirement already satisfied: dm-tree in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-datasets) (0.1.9)
Requirement already satisfied: etils>=1.9.1 in c:\users\user\appdata\roaming\python\python312\site-packages (from etils[edc,enp,epath,epy,etree]>=1.
9.1; python_version >= "3.11"->tensorflow-datasets) (1.12.2)
Requirement already satisfied: immutabledict in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-datasets) (4.2.1)
Requirement already satisfied: promise in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-datasets) (2.3)
Requirement already satisfied: protobuf>=3.20 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-datasets) (4.25.3)
Requirement already satisfied: psutil in c:\programdata\anaconda3\lib\site-packages (from tensorflow-datasets) (5.9.0)
Requirement already satisfied: pyarrow in c:\programdata\anaconda3\lib\site-packages (from tensorflow-datasets) (16.1.0)
Requirement already satisfied: requests>=2.19.0 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-datasets) (2.32.3)
Requirement already satisfied: simple_parsing in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-datasets) (0.1.7)
Requirement already satisfied: tensorflow-metadata in c:\users\user\appdata\roaming\python\python312\site-packages (from tensorflow-datasets) (1.17.
```

[3]:
```
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os
from pathlib import Path
```

[4]:
```
tfds.disable_progress_bar()
```

[7]:
```
dataset, info = tfds.load('oxford_flowers102', with_info=True, as_supervised=True)
train_ds, val_ds, test_ds = dataset['train'], dataset['validation'], dataset['test']
```

[8]:
```
print("Dataset Name:", info.name)
```

```
Dataset Name: oxford_flowers102
```

[9]:
```
print("Number of classes:", info.features['label'].num_classes)
```

```
Number of classes: 102
```

[10]:
```
print("Total examples:")
print(" - Train:", info.splits['train'].num_examples)
print(" - Validation:", info.splits['validation'].num_examples)
print(" - Test:", info.splits['test'].num_examples)
```

```
Total examples:
 - Train: 1020
 - Validation: 1020
 - Test: 6149
```

[11]:
```
label_names = info.features['label'].names
```

[49]:
```
label_names = info.features['label'].names
```

[14]:
```
res_df = pd.DataFrame(image_shapes, columns=['Height', 'Width', 'Channels'])
print("Image resolution statistics (100 samples):")
print(res_df.describe())
```

```
Image resolution statistics (100 samples):
           Height       Width  Channels
count  100.000000  100.000000     100.0
mean   541.670000  623.080000       3.0
std     78.265806  106.863046       0.0
min    500.000000  500.000000       3.0
25%    500.000000  500.000000       3.0
50%    500.000000  660.500000       3.0
75%    544.250000  726.750000       3.0
max    883.000000  825.000000       3.0
```

[16]:
```
print(f"Number of classes: {info.features['label'].num_classes}")
```

```
Number of classes: 102
```

[36]:
```
plt.figure(figsize=(12, 10))
for i, (img, lbl) in enumerate(train_ds.take(9)):
    plt.subplot(3, 3, i+1)
    plt.imshow(img)
    plt.axis('off')
    plt.title(label_names[lbl.numpy()])
plt.tight_layout()
plt.show()
```

[29]:
```
OUTPUT_DIR = Path("images_and_statistics")
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)
```

[30]:
```
folder_name = "images_and_statistics"
if not os.path.exists(folder_name):
    os.makedirs(folder_name)
    print(f"✅ Folder '{folder_name}' created.")
else:
    print(f"📁 Folder '{folder_name}' already exists.")
```

```
📁 Folder 'images_and_statistics' already exists.
```

[31]:
```
plt.savefig("images_and_statistics/sample_plot.png")
```

```
<Figure size 640x480 with 0 Axes>
```

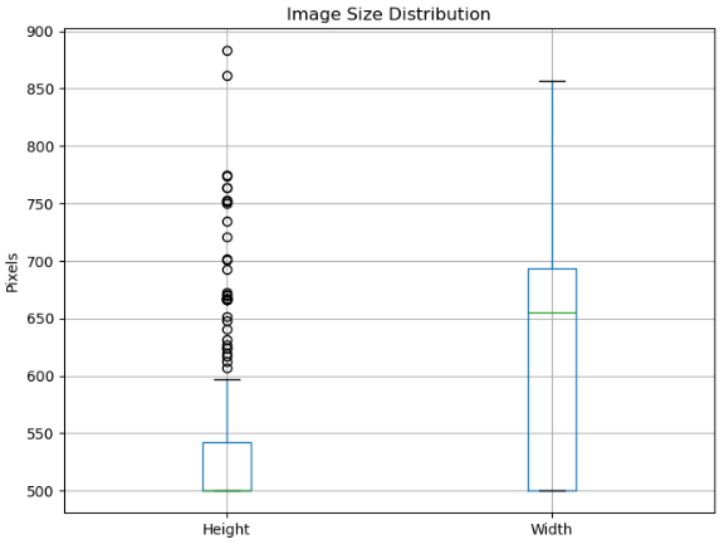
[34]:
```
ds = tfds.load('oxford_flowers102', split='train', as_supervised=True)
os.makedirs("images_and_statistics", exist_ok=True)
for image, label in ds.take(1):
    plt.figure(figsize=(4, 4))
    plt.imshow(image.numpy())
    plt.axis('off')
    plt.title(f"Flower Class: {label.numpy()}")
    plt.savefig("images_and_statistics/sample_flower.png")
    plt.show()
```

[35]:
```
os.makedirs("images_and_statistics", exist_ok=True)
dataset = tfds.load('oxford_flowers102', split='train', as_supervised=True)
plt.figure(figsize=(10, 10))
for i, (image, label) in enumerate(dataset.take(9)):
    plt.subplot(3, 3, i + 1)
    plt.imshow(image.numpy())
    plt.title(f"Class ID: {label.numpy()}")
    plt.axis("off")
plt.tight_layout()
plt.savefig("images_and_statistics/flower_grid.png")
plt.show()
```

[44]:
```
output_dir = "images_and_statistics"
os.makedirs(output_dir, exist_ok=True)
dataset = tfds.load("oxford_flowers102", split="train", as_supervised=True)
image_sizes = []
for image, label in dataset.take(200):  # Analyze first 200 images
    h, w = image.shape[0], image.shape[1]
    image_sizes.append((h, w))
```

[45]:
```
df_sizes = pd.DataFrame(image_sizes, columns=["Height", "Width"])
csv_path = os.path.join(output_dir, "image_size_stats.csv")
df_sizes.to_csv(csv_path, index=False)
```

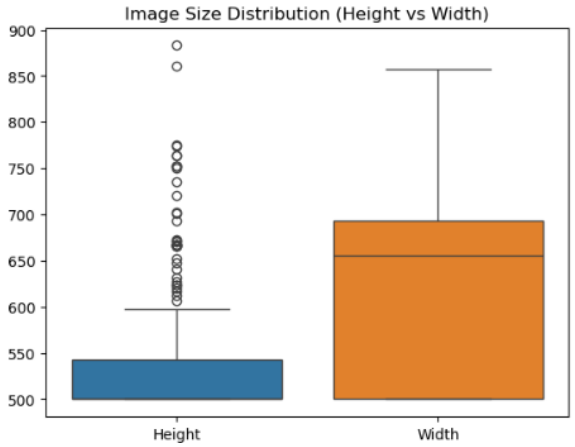
[46]:
```
plt.figure(figsize=(8, 6))
df_sizes.boxplot()
plt.title("Image Size Distribution")
plt.ylabel("Pixels")
plt.grid(True)
boxplot_path = os.path.join(output_dir, "image_size_boxplot.png")
plt.savefig(boxplot_path)
plt.show()
```



[47]:
```
plt.figure(figsize=(10, 10))
for i, (image, label) in enumerate(dataset.take(9)):
    plt.subplot(3, 3, i + 1)
    plt.imshow(image.numpy())
    plt.title(f"Class ID: {label.numpy()}")
    plt.axis("off")
```

[1]:
```python
import tensorflow_datasets as tfds
import pandas as pd

# Load dataset (as supervised gives (image, label) tuple)
ds, ds_info = tfds.load("oxford_flowers102", split='train', as_supervised=True, with_info=True)

# Sample 200 images and collect dimensions
image_sizes = []

for i, (image, label) in enumerate(ds.take(200)):
    height = image.shape[0]
    width = image.shape[1]
    image_sizes.append({'Index': i+1, 'Height': height, 'Width': width})

# Convert to DataFrame
image_sizes_df = pd.DataFrame(image_sizes)

# Save to CSV
image_sizes_df.to_csv('images_and_statistics/flower_image_stats.csv', index=False)
```

[2]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('images_and_statistics/flower_image_stats.csv')
sns.boxplot(data=df[['Height', 'Width']])
plt.title("Image Size Distribution (Height vs Width)")
plt.savefig("images_and_statistics/image_size_boxplot.png")
```



File View Settings Help

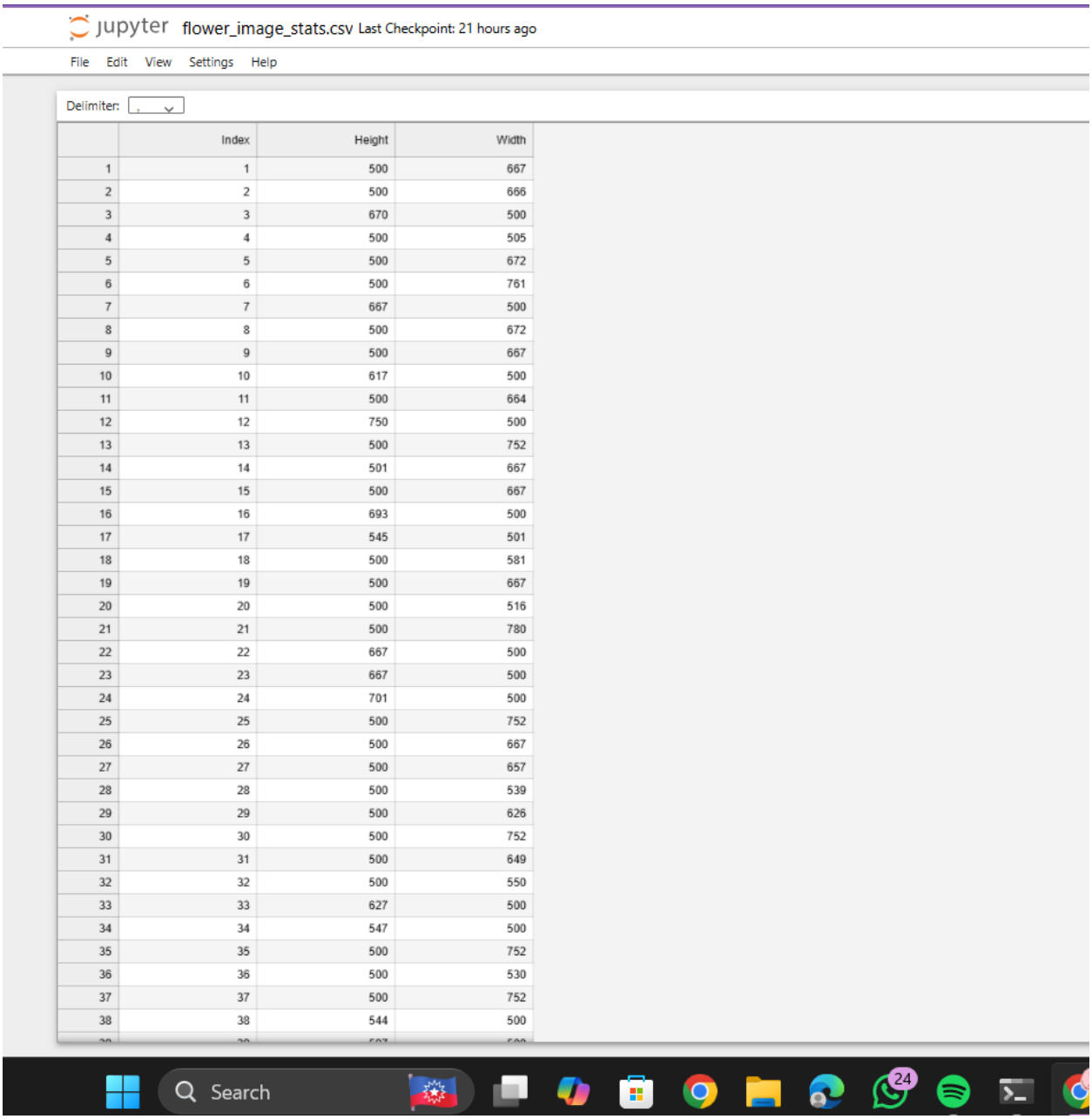
Files Running

Select items to perform actions on them.

/ NULLCLASS-INTERNSHIP-TASKS / Task2-Dataset-Exploration / images_and_statistics /

Name

- flower_grid.png
- flower_image_stats.csv
- image_size_boxplot.png
- image_size_stats.csv
- sample_flower.png



	Index	Height	Width
1	1	500	667
2	2	500	666
3	3	670	500
4	4	500	505
5	5	500	672
6	6	500	761
7	7	667	500
8	8	500	672
9	9	500	667
10	10	617	500
11	11	500	664
12	12	750	500
13	13	500	752
14	14	501	667
15	15	500	667
16	16	693	500
17	17	545	501
18	18	500	581
19	19	500	667
20	20	500	516
21	21	500	780
22	22	667	500
23	23	667	500
24	24	701	500
25	25	500	752
26	26	500	667
27	27	500	657
28	28	500	539
29	29	500	626
30	30	500	752
31	31	500	649
32	32	500	550
33	33	627	500
34	34	547	500
35	35	500	752
36	36	500	530
37	37	500	752
38	38	544	500

TASK 3

[1]:
```
import os
import csv
from PIL import Image
```

[2]:
```
dataset_dir = "dataset"
images_dir = os.path.join(dataset_dir, "images")
os.makedirs(images_dir, exist_ok=True)
```

[3]:
```
img_names = ["img1.jpg", "img2.jpg", "medical_scan_01.jpg"]
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255)]  # Red, Green, Blue

for name, color in zip(img_names, colors):
    img = Image.new("RGB", (512, 512), color)
    img.save(os.path.join(images_dir, name))
```

[4]:
```
captions = [
    ("img1.jpg", "A red flower in a green field"),
    ("img2.jpg", "A futuristic medical scanner in a lab"),
    ("medical_scan_01.jpg", "High-resolution MRI scan showing brain activity"),
]
csv_path = os.path.join(dataset_dir, "captions.csv")
with open(csv_path, "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["file_name", "text"])
    writer.writerows(captions)

print("Dataset prepared!")
```

```
Dataset prepared!
```

[5]:
```
import torch
import numpy as np
import pandas as pd
```

[6]:
```
from torch.utils.data import Dataset, DataLoader
from transformers import CLIPTokenizer, CLIPTextModel
from diffusers import AutoencoderKL, UNet2DConditionModel, StableDiffusionPipeline, DDIMScheduler
```

[7]:
```
from accelerate import Accelerator
from tqdm import tqdm
```

[8]:
```python
class CustomImageDataset(Dataset):
    def __init__(self, images_dir, captions_file, tokenizer, size=512):
        self.images_dir = images_dir
        self.data = pd.read_csv(captions_file)
        self.tokenizer = tokenizer
        self.size = size

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        row = self.data.iloc[idx]
        image_path = os.path.join(self.images_dir, row['file_name'])
        image = Image.open(image_path).convert("RGB").resize((self.size, self.size))
        image = torch.tensor(np.array(image)).permute(2, 0, 1).float() / 255.0

        inputs = self.tokenizer(
            row['text'],
            truncation=True,
            padding="max_length",
            max_length=self.tokenizer.model_max_length,
            return_tensors="pt"
        )

        return {
            "pixel_values": image,
            "input_ids": inputs.input_ids.squeeze(0)
        }
```

[9]:
```python
PRETRAINED_MODEL_NAME = "CompVis/stable-diffusion-v1-4"
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
SEED = 42
torch.manual_seed(SEED)
```

[9]:
```
<torch._C.Generator at 0x18553fffb90>
```

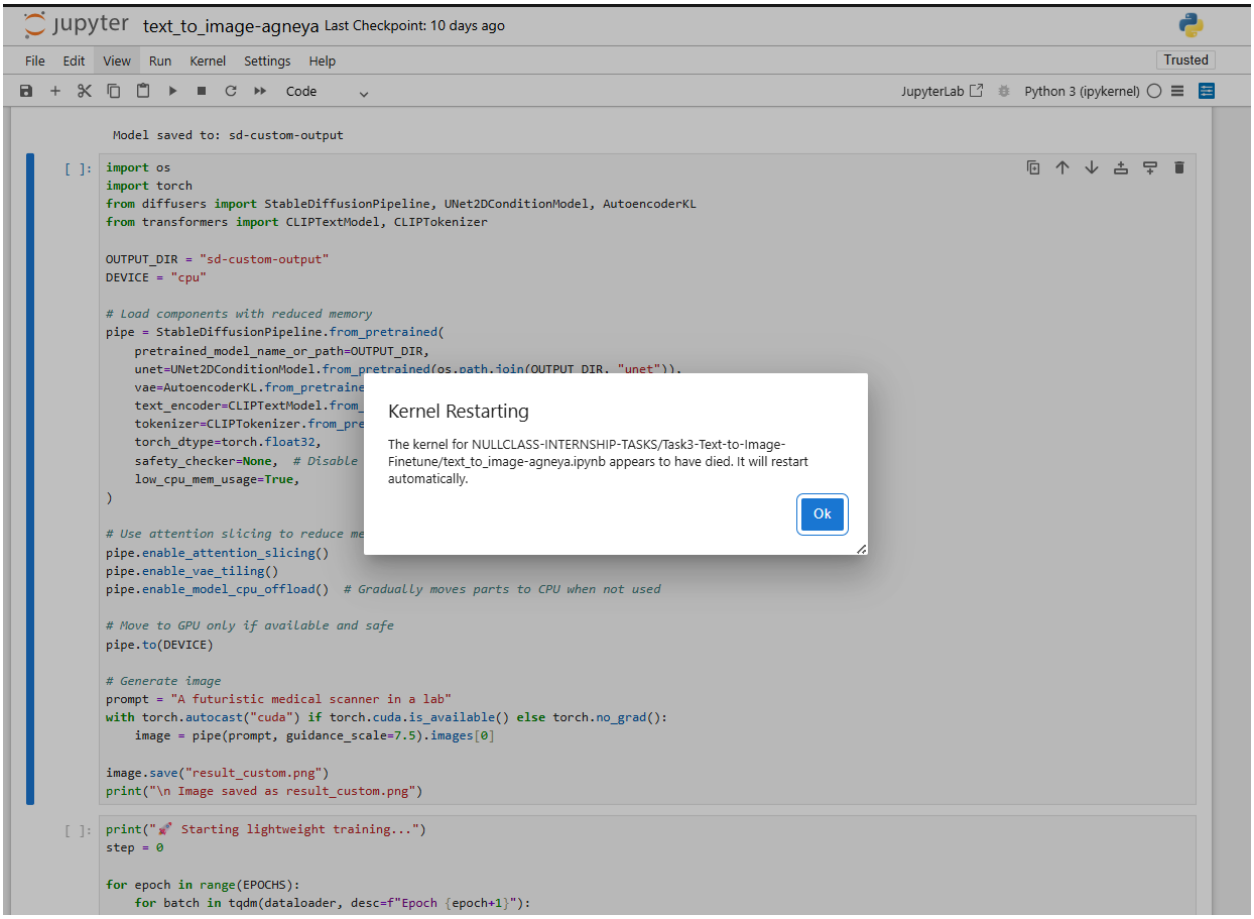
[10]:
```python
tokenizer = CLIPTokenizer.from_pretrained(PRETRAINED_MODEL_NAME, subfolder="tokenizer")
text_encoder = CLIPTextModel.from_pretrained(PRETRAINED_MODEL_NAME, subfolder="text_encoder").to(DEVICE)
vae = AutoencoderKL.from_pretrained(PRETRAINED_MODEL_NAME, subfolder="vae").to(DEVICE)
unet = UNet2DConditionModel.from_pretrained(PRETRAINED_MODEL_NAME, subfolder="unet").to(DEVICE)

dataset = CustomImageDataset(
    images_dir=images_dir,
    captions_file=csv_path,
    tokenizer=tokenizer
)
dataloader = DataLoader(dataset, batch_size=2, shuffle=True)
```

[13]:
```python
print("Starting training...")
step = 0
MAX_TRAIN_STEPS = 10

for epoch in range(1):
    for batch in tqdm(dataloader, desc=f"Epoch {epoch+1}"):
        pixel_values = batch["pixel_values"].to(DEVICE)
        input_ids = batch["input_ids"].to(DEVICE)

        with torch.no_grad():
            encoder_hidden_states = text_encoder(input_ids)[0]
            latents = vae.encode(pixel_values).latent_dist.sample() * 0.18215

        noise = torch.randn_like(latents)
        timesteps = torch.randint(0, 1000, (latents.shape[0],), device=DEVICE).long()
        noisy_latents = noise_scheduler.add_noise(latents, noise, timesteps)

        noise_pred = unet(noisy_latents, timesteps, encoder_hidden_states).sample
        loss = torch.nn.functional.mse_loss(noise_pred, noise)

        optimizer.zero_grad()
        accelerator.backward(loss)
        optimizer.step()

        if step % 5 == 0:
            print(f"Step {step} | Loss: {loss.item():.4f}")

        step += 1
        torch.cuda.empty_cache()

        if step >= MAX_TRAIN_STEPS:
            break

    if step >= MAX_TRAIN_STEPS:
        print("\n Training complete.")
        break
```

```
Starting training...
Epoch 1:  50%|████████████████████████████████                                | 1/2 [04:09<04:09, 249.63s/it]
Step 0 | Loss: 0.0043
Epoch 1: 100%|████████████████████████████████████████████████████████████████| 2/2 [07:37<00:00, 228.65s/it]
```

[14]:
```python
# Save model
OUTPUT_DIR = "sd-custom-output"
if accelerator.is_main_process:
    os.makedirs(OUTPUT_DIR, exist_ok=True)
    unet.save_pretrained(os.path.join(OUTPUT_DIR, "unet"))
    vae.save_pretrained(os.path.join(OUTPUT_DIR, "vae"))
    text_encoder.save_pretrained(os.path.join(OUTPUT_DIR, "text_encoder"))
    tokenizer.save_pretrained(os.path.join(OUTPUT_DIR, "tokenizer"))
    print(f"\n Model saved to: {OUTPUT_DIR}")
```

```
Model saved to: sd-custom-output
```



Full trained pipeline or sd-custom-output folder is heavy , Jupyter Notebook kernel keeps restarting.

The model or pipeline is **too large to load fully into your system's RAM/VRAM**.

You're hitting **memory overflow**, especially during:

- StableDiffusionPipeline.from_pretrained(...)
- Running pipe(prompt) for image generation.