```
!pip install diffusers transformers accelerate torch safetensors
```

```
Requirement already satisfied: diffusers in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: safetensors in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: importlib_metadata in /usr/local/lib/python3.12/d
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: httpx<1.0.0 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: huggingface-hub<2.0,>=0.34.0 in /usr/local/lib/py
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/di
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: Pillow in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/pyt
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/pytho
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: fsspec>=0.8.5 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/d
```

```
import torch
from diffusers import StableDiffusionPipeline
import os
```

```
Flax classes are deprecated and will be removed in Diffusers v1.0.0. We recommen
Flax classes are deprecated and will be removed in Diffusers v1.0.0. We recommen
```

Task 1

```
model_id = "runwayml/stable-diffusion-v1-5"
```

```
pipe = StableDiffusionPipeline.from_pretrained(
    model_id,
    torch_dtype=torch.float16
)

pipe = pipe.to("cuda")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: Userw
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access publ
  warnings.warn(
model_index.json: 100%  541/541 [00:00<00:00, 20.9kB/s]
Fetching 15 files: 100%  15/15 [00:53<00:00, 3.88s/it]
preprocessor_config.json: 100%  342/342 [00:00<00:00, 15.7kB/s]
config.json: 100%  617/617 [00:00<00:00, 15.0kB/s]
special_tokens_map.json: 100%  472/472 [00:00<00:00, 6.86kB/s]
config.json: 4.72k/? [00:00<00:00, 63.8kB/s]
scheduler_config.json: 100%  308/308 [00:00<00:00, 5.50kB/s]
merges.txt: 525k/? [00:00<00:00, 10.3MB/s]
text_encoder/model.safetensors: 100%  492M/492M [00:38<00:00, 83.8MB/s]
safety_checker/model.safetensors: 100%  1.22G/1.22G [00:22<00:00, 27.1MB/s]
tokenizer_config.json: 100%  806/806 [00:00<00:00, 67.2kB/s]
vocab.json: 1.06M/? [00:00<00:00, 32.3MB/s]
config.json: 100%  547/547 [00:00<00:00, 48.4kB/s]
config.json: 100%  743/743 [00:00<00:00, 47.2kB/s]
unet/diffusion_pytorch_model.safetensors: 100%  3.44G/3.44G [00:52<00:00, 44.5MB/s]
vae/diffusion_pytorch_model.safetensors: 100%  335M/335M [00:43<00:00, 8.92MB/s]
Loading pipeline components...: 100%  7/7 [00:20<00:00, 3.55s/it]
`torch_dtype` is deprecated! Use `dtype` instead!
```

```
prompts = [
    "A futuristic city at night with neon lights",
    "A peaceful mountain landscape during sunrise",
    "A robot studying in a classroom",
    "A realistic portrait of a medieval warrior",
```

```
    "A cyberpunk street with rain and reflections"
]
```

```
output_dir = "synthetic_dataset"
os.makedirs(output_dir, exist_ok=True)

for idx, prompt in enumerate(prompts):
    image = pipe(prompt).images[0]
    image.save(f"{output_dir}/image_{idx+1}.png")

print("Synthetic dataset generated successfully.")
```

```
100%  50/50 [00:10<00:00,  6.91it/s]
100%  50/50 [00:08<00:00,  6.33it/s]
100%  50/50 [00:07<00:00,  5.62it/s]
100%  50/50 [00:08<00:00,  6.63it/s]
100%  50/50 [00:07<00:00,  6.44it/s]
Synthetic dataset generated successfully.
```

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("synthetic_dataset/image_1.png")
plt.imshow(img)
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("synthetic_dataset/image_2.png")
plt.imshow(img)
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("synthetic_dataset/image_3.png")
plt.imshow(img)
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("synthetic_dataset/image_4.png")
plt.imshow(img)
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
from PIL import Image
import matplotlib.pyplot as plt

img = Image.open("synthetic_dataset/image_5.png")
plt.imshow(img)
plt.axis("off")
```

```
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
Start coding or generate with AI.
```

Task 2

```
BASE_DIR = "synthetic_chest_xray_dataset_task2"
os.makedirs(BASE_DIR, exist_ok=True)
```

```
dataset_labels = {
    "normal_anatomy": "healthy human lungs with normal anatomy chest X-ray",
    "infectious_patterns": "chest X-ray showing infectious lung disease such as
    "lung_opacities": "chest X-ray showing lung opacities including ground glas
    "pleural_conditions": "chest X-ray showing pleural effusion or pneumothorax
    "structural_lesions": "chest X-ray showing lung nodules masses or fibrosis"
    "cardiac_findings": "chest X-ray showing cardiomegaly and pulmonary vascula
    "medical_devices": "chest X-ray showing medical devices such as tubes cathe
    "imaging_artifacts": "chest X-ray with imaging artifacts such as noise moti
    "view_positioning": "chest X-ray with different views and patient positioni
    "domain_shift": "chest X-ray showing domain shift due to different scanners
}
```

```
BASE_PROMPT = (
    "Very high quality realistic chest X-ray showing {}, "
    "medical radiology style, grayscale, diagnostic accuracy, "
    "sharp details, hospital imaging"
```

```
)
```

```
IMAGES_PER_CATEGORY = 5

for category, label in dataset_labels.items():
    category_path = os.path.join(BASE_DIR, category)
    os.makedirs(category_path, exist_ok=True)

    prompt = BASE_PROMPT.format(label)
    print(f"Generating images for {category}")

    for i in range(IMAGES_PER_CATEGORY):
        image = pipe(
            prompt,
            guidance_scale=8.5,
            num_inference_steps=40
        ).images[0]

        image.save(f"{category_path}/{category}_{i+1}.png")

print("TASK-2 synthetic dataset generation completed.")
```

```
Generating images for normal_anatomy
100%                                                 40/40 [00:06<00:00,  5.72it/s]
100%                                                 40/40 [00:10<00:00,  4.48it/s]
100%                                                 40/40 [00:06<00:00,  6.31it/s]
100%                                                 40/40 [00:06<00:00,  6.57it/s]
100%                                                 40/40 [00:06<00:00,  6.72it/s]
Generating images for infectious_patterns
100%                                                 40/40 [00:06<00:00,  6.80it/s]
100%                                                 40/40 [00:05<00:00,  6.91it/s]
100%                                                 40/40 [00:05<00:00,  6.91it/s]
100%                                                 40/40 [00:05<00:00,  6.94it/s]
100%                                                 40/40 [00:05<00:00,  6.92it/s]
Generating images for lung_opacities
100%                                                 40/40 [00:05<00:00,  6.85it/s]
100%                                                 40/40 [00:06<00:00,  6.82it/s]
100%                                                 40/40 [00:06<00:00,  6.68it/s]
100%                                                 40/40 [00:06<00:00,  6.58it/s]
100%                                                 40/40 [00:06<00:00,  6.54it/s]
Generating images for pleural_conditions
100%                                                 40/40 [00:06<00:00,  6.54it/s]
100%                                                 40/40 [00:06<00:00,  6.59it/s]
100%                                                 40/40 [00:06<00:00,  6.65it/s]
100%                                                 40/40 [00:06<00:00,  6.68it/s]
100%                                                 40/40 [00:06<00:00,  6.77it/s]
Generating images for structural_lesions
100%                                                 40/40 [00:06<00:00,  6.78it/s]
100%                                                 40/40 [00:06<00:00,  6.83it/s]
100%                                                 40/40 [00:06<00:00,  6.79it/s]
```

```
sample_image = Image.open(
    f"{BASE_DIR}/normal_anatomy/normal_anatomy_1.png"
)

plt.imshow(sample_image)
plt.axis("off")
```

```
100%                                                 40/40 [00:06<00:00,  6.70it/s]
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
[00:06<00:00,  6.74it/s]
[00:06<00:00,  6.63it/s]
[00:06<00:00,  6.67it/s]
[00:06<00:00,  6.74it/s]
[00:06<00:00,  6.70it/s]
[00:06<00:00,  6.74it/s]
[00:06<00:00,  6.73it/s]
[00:06<00:00,  6.78it/s]
[00:06<00:00,  6.73it/s]
[00:06<00:00,  6.77it/s]
[00:06<00:00,  6.75it/s]
100%                                                 40/40 [00:06<00:00,  6.69it/s]
```

Start coding or generate with AI.

```
100%                                                 40/40 [00:06<00:00,  6.66it/s]
```

TASK 3

```
100%                                                 40/40 [00:06<00:00,  6.67it/s]
```

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

```
model = models.densenet121(pretrained=True)

num_features = model.classifier.in_features
model.classifier = torch.nn.Linear(num_features, 2)  # Normal vs Abnormal
model.eval()
```

```
      (denselayer2): _DenseLayer(
        (norm1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(96, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      )
      (denselayer3): _DenseLayer(
        (norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      )
      (denselayer4): _DenseLayer(
        (norm1): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(160, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      )
      (denselayer5): _DenseLayer(
        (norm1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      )
      (denselayer6): _DenseLayer(
```

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
```