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

```
Requirement already satisfied: diffusers in /usr/local/lib/python3.12/dist-packages (0.36.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.57.3)
Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-packages (1.12.0)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.9.0+cpu)
Requirement already satisfied: safetensors in /usr/local/lib/python3.12/dist-packages (0.7.0)
Requirement already satisfied: importlib_metadata in /usr/local/lib/python3.12/dist-packages (from diffusers) (8.7.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from diffusers) (3.20.2)
Requirement already satisfied: httpx<1.0.0 in /usr/local/lib/python3.12/dist-packages (from diffusers) (0.28.1)
Requirement already satisfied: huggingface-hub<2.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from diffusers) (0.3
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from diffusers) (2.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from diffusers) (2025.11.3)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from diffusers) (2.32.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.12/dist-packages (from diffusers) (11.3.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.14.0)
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch) (3.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec>=0.8.5 in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->diffusers) (4.12.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->diffusers) (2026.1.4)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->diffusers) (1.0.9
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0->diffusers) (3.11)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1.0.0->diffus
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<2.0,>=0
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch) (1.
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.12/dist-packages (from importlib_metadata->diffusers) (3
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.3)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->diffusers
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->diffusers) (2.5
```

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

```
Flax classes are deprecated and will be removed in Diffusers v1.0.0. We recommend migrating to PyTorch classes or pinning yo
Flax classes are deprecated and will be removed in Diffusers v1.0.0. We recommend migrating to PyTorch classes or pinning yo
```

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: UserWarning:
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 (https://huggingface.co/settings/tokens), set
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  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!
```

```python
prompts = [
    "Ultra-detailed futuristic megacity at night, neon holograms, flying vehicles, rain, cinematic lighting, cyberpunk aest

    "Photorealistic peaceful mountain valley at sunrise, golden hour light, misty atmosphere, pine trees, high dynamic rang

    "Friendly humanoid robot studying in a modern classroom, books and holographic screen, soft lighting, realistic texture

    "Hyper-realistic portrait of a medieval warrior wearing detailed armor, battle scars, dramatic lighting, shallow depth

    "Cyberpunk street scene at night with heavy rain, neon reflections on wet pavement, moody atmosphere, cinematic compos
]
```

```python
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.
```

```python
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")
```

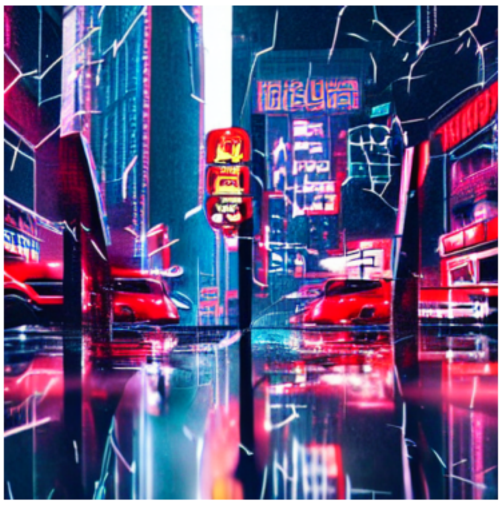
```
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
```

```
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": "normal healthy adult lungs with clear lung fields and normal cardiomediastinal silhouette",

    "infectious_patterns": "infectious lung disease such as bacterial or viral pneumonia with patchy infiltrates",

    "lung_opacities": "bilateral lung opacities including ground glass opacities and focal consolidations",

    "pleural_conditions": "pleural effusion or pneumothorax with visible pleural abnormalities",

    "structural_lesions": "lung nodules masses fibrotic changes or interstitial lung disease patterns",

    "cardiac_findings": "cardiomegaly with enlarged cardiac silhouette and pulmonary vascular congestion",
    "medical_devices": "presence of medical devices including endotracheal tube central venous catheter or pacemaker leads'

    "imaging_artifacts": "radiographic imaging artifacts such as motion blur noise grid artifacts or exposure imbalance",

    "view_positioning": "different chest X-ray views including PA AP supine and erect positioning",

    "domain_shift": "appearance variations due to different scanners hospitals imaging protocols and resolutions"
}
```

```
BASE_PROMPT = (
    "Highly realistic diagnostic chest X-ray image showing {}, "
    "true radiology appearance, grayscale only, frontal chest radiograph, "
    "accurate anatomy, clinical quality, sharp contrast, hospital PACS style"
)
```

```
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]
```

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

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

```
100%                                   40/40 [00:05<00:00,  6.85it/s]
(np.float64(-0.5), np.float64(511.5), np.float64(511.5), np.float64(-0.5))
                                             [00:06<00:00,  6.82it/s]
                                             [00:06<00:00,  6.68it/s]
                                             [00:06<00:00,  6.58it/s]
                                             [00:06<00:00,  6.54it/s]
                                             [00:06<00:00,  6.54it/s]
                                             [00:06<00:00,  6.59it/s]
                                             [00:06<00:00,  6.65it/s]
                                             [00:06<00:00,  6.68it/s]
                                             [00:06<00:00,  6.77it/s]
                                             [00:06<00:00,  6.78it/s]
                                             [00:06<00:00,  6.83it/s]
100%                                   40/40 [00:06<00:00,  6.79it/s]
```

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

```
100%                                   40/40 [00:06<00:00,  6.80it/s]
Generating images for cardiac_findings
100%                                   40/40 [00:06<00:00,  6.76it/s]
```

TASK 3

```
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()
```

```
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is depre
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum
  warnings.warn(msg)
DenseNet(
  (features): Sequential(
    (conv0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (norm0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu0): ReLU(inplace=True)
    (pool0): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (denseblock1): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(64, 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)
      )
      (denselayer2): _DenseLayer(
100%  40/40  [00:06<00:00,   6.65it/s]
Generating images for domain shift
100%  40/40  [00:06<00:00,   6.72it/s]
100%  40/40  [00:06<00:00,   6.77it/s]
100%  40/40  [00:06<00:00,   6.79it/s]
100%  40/40  [00:06<00:00,   6.77it/s]
100%  40/40  [00:06<00:00,   6.74it/s]
TASK-2 synthetic dataset generation completed.
```

```
      (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(
      (norm1): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(224, 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)
```

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

```python
DATASET_DIR = "synthetic_chest_xray_dataset_task2"
```

```python
def get_true_label(folder_name):
    if folder_name == "normal_anatomy":
        return 0  # Normal
    else:
        return 1  # Abnormal
```

```python
correct = 0
total = 0

print("Image | Ground Truth | Predicted")
print("----------------------------------")

for category in os.listdir(DATASET_DIR):
    category_path = os.path.join(DATASET_DIR, category)

    if not os.path.isdir(category_path):
        continue

    true_label = get_true_label(category)

    for img_name in os.listdir(category_path):
        img_path = os.path.join(category_path, img_name)

        image = Image.open(img_path).convert("RGB")
        input_tensor = transform(image).unsqueeze(0)

        with torch.no_grad():
            output = model(input_tensor)
            predicted_label = torch.argmax(output, dim=1).item()

        print(f"{img_name} | {true_label} | {predicted_label}")
```