```
pip install torch torchvision matplotlib pillow numpy
```

```
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 31.1 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 46.7 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 1.2 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 4.2 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 15.9 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 6.6 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 4.3 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 35.0 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cud
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-ru
```

```
import os
import matplotlib.pyplot as plt
from PIL import Image
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms

# ---------- STEP 1: DATASET DEFINITION ----------
class XRayDataset(Dataset):
    def __init__(self, root, transform=None, max_per_class=None):
        self.samples = []
        self.transform = transform
        for label, cls in enumerate(["NORMAL", "PNEUMONIA"]):
            cls_path = os.path.join(root, cls)
            files = [f for f in os.listdir(cls_path) if f.endswith((".jpeg", ".jpg", ".png"))]
            files = sorted(files)[:max_per_class] if max_per_class else files
            for f in files:
```

```
                    self.samples.append((os.path.join(cls_path, f), label))

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

        def __getitem__(self, idx):
            path, label = self.samples[idx]
            img = Image.open(path).convert("L")
            if self.transform:
                img = self.transform(img)
            return img, label

    # ---------- STEP 2: SIMPLE CNN MODEL ----------
    class SimpleCNN(nn.Module):
        def __init__(self):
            super().__init__()
            self.conv1 = nn.Conv2d(1, 8, 3, padding=1)
            self.conv2 = nn.Conv2d(8, 16, 3, padding=1)
            self.fc1 = nn.Linear(16 * 32 * 32, 64)
            self.fc2 = nn.Linear(64, 2)

        def forward(self, x):
            x = F.relu(F.max_pool2d(self.conv1(x), 2))
            x = F.relu(F.max_pool2d(self.conv2(x), 2))
            x = x.view(x.size(0), -1)
            x = F.relu(self.fc1(x))
            return self.fc2(x)

    # ---------- STEP 3: TRAINING FUNCTION ----------
    def train_model():
        print("📊 Preparing data...")
        train_transform = transforms.Compose([
            transforms.Resize((128, 128)),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
        ])

        # Load and limit samples
        full_ds = XRayDataset("chest_xray/train", transform=train_transform)
        normal_samples = [s for s in full_ds.samples if s[1] == 0][:50]
        pneumonia_samples = [s for s in full_ds.samples if s[1] == 1][:50]
        combined_samples = normal_samples + pneumonia_samples

        # Custom dataset from combined samples
        class SubsetDataset(Dataset):
            def __init__(self, samples, transform):
                self.samples = samples
                self.transform = transform

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

            def __getitem__(self, idx):
                path, label = self.samples[idx]
                img = Image.open(path).convert("L")
                if self.transform:
                    img = self.transform(img)
                return img, label

        train_ds = SubsetDataset(combined_samples, train_transform)
        train_loader = DataLoader(train_ds, batch_size=32, shuffle=True)

        model = SimpleCNN()
        optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
        loss_fn = nn.CrossEntropyLoss()

        model.train()
        for epoch in range(5):
            total_loss = 0
            for xb, yb in train_loader:
                optimizer.zero_grad()
                out = model(xb)
                loss = loss_fn(out, yb)
                loss.backward()
                optimizer.step()
                total_loss += loss.item()
            print(f"✅ Epoch {epoch+1}: Loss = {total_loss / len(train_loader):.4f}")
```

```
    return model, train_transform

    print(f"✅ Displayed {shown_normal} NORMAL and {shown_pneumonia} PNEUMONIA examples.")

# ---------- RUN PIPELINE ----------
if __name__ == "__main__":
    model, tf = train_model()
```

```
📊 Preparing data...
✅ Epoch 1: Loss = 0.7006
✅ Epoch 2: Loss = 0.6947
✅ Epoch 3: Loss = 0.6982
✅ Epoch 4: Loss = 0.6938
✅ Epoch 5: Loss = 0.6938
```

```
# ---------- STEP 5: INFERENCE ----------
def predict(model, transform):
    model.eval()
    test_ds = XRayDataset("chest_xray/test", transform=transform)

    shown_normal = 0
    shown_pneumonia = 0
    max_to_show = 3  # Show 3 examples per class
    shown = 0

    for i in range(len(test_ds)):
        if shown_normal >= max_to_show and shown_pneumonia >= max_to_show:
            break

        img, label = test_ds[i]
        x = img.unsqueeze(0)

        with torch.no_grad():
            pred = model(x).argmax(1).item()

        is_normal = (label == 0)
        if is_normal and shown_normal < max_to_show:
            shown_normal += 1
        elif not is_normal and shown_pneumonia < max_to_show:
            shown_pneumonia += 1
        else:
            continue  # Skip if we've already shown enough from this class

        title = f"True: {'PNEUMONIA' if label else 'NORMAL'} | Predicted: {'PNEUMONIA' if pred else 'NORMAL'}"
        img_np = img.squeeze().numpy()
        plt.imshow(img_np, cmap='gray')
        plt.title(title)
        plt.axis('off')
        plt.show()
        shown += 1

    print(f"✅ Displayed {shown_normal} NORMAL and {shown_pneumonia} PNEUMONIA examples.")
if __name__ == "__main__":
    predict(model, tf)
```

True: NORMAL | Predicted: PNEUMONIA

True: NORMAL | Predicted: PNEUMONIA

True: NORMAL | Predicted: PNEUMONIA