

## CODE FOR MULTIMODAL AI DIAGNOSIS:

```
import torch
import torchvision.transforms as transforms
from PIL import Image
import numpy as np
import gradio as gr
import torchxrayvision as xrv
import matplotlib.pyplot as plt
import cv2
import io
from torchvision import models
import torch.nn.functional as F

chexnet_model = xrv.models.DenseNet(weights="densenet121-res224-all")
chexnet_model.eval()
chexnet_labels = chexnet_model.pathologies

brain_model =
models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
brain_model.fc = torch.nn.Linear(brain_model.fc.in_features, 2)
brain_model.eval()
brain_labels = ["No Tumor", "Tumor"]

knee_labels = ["Normal", "Osteoarthritis"]

def preprocess_chest(image):
    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Lambda(lambda x: x[0:1, :, :]),
        transforms.Normalize([0.5], [0.5])
    ])
    return transform(image).unsqueeze(0)

def preprocess_brain(image):
    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
    return transform(image.convert("RGB")).unsqueeze(0)

def diagnose_with_cam(image, body_part):
    try:
        if body_part == "Chest X-ray":
            img_tensor = preprocess_chest(image)
```

```

img_tensor.requires_grad = True
with torch.no_grad():
    preds = chexnet_model(img_tensor)[0].sigmoid().numpy()
    top_indices = np.argsort(preds)[-3:][::-1]
    result = "\n".join([f"{chexnet_labels[i]}: {preds[i]:.2f}" for i in
top_indices])

```

```

class GradCAM:

```

```

    def __init__(self, model, target_layer):
        self.model = model
        self.target_layer = target_layer
        self.gradients = None
        self.activations = None
        target_layer.register_forward_hook(self.forward_hook)
        target_layer.register_backward_hook(self.backward_hook)

```

```

    def forward_hook(self, module, input, output):
        self.activations = output.detach()

```

```

    def backward_hook(self, module, grad_input, grad_output):
        self.gradients = grad_output[0].detach()

```

```

    def generate_heatmap(self, input_tensor, class_idx):
        output = self.model(input_tensor)
        self.model.zero_grad()
        class_score = output[0][class_idx]
        class_score.backward(retain_graph=True)
        weights = self.gradients.mean(dim=[2, 3], keepdim=True)
        cam = (weights * self.activations).sum(dim=1, keepdim=True)
        cam = torch.relu(cam)
        cam = cam.squeeze().numpy()
        cam = (cam - cam.min()) / (cam.max() - cam.min())
        cam = cv2.resize(cam, (224, 224))
        return cam

```

```

    cam = GradCAM(chexnet_model,
chexnet_model.features.denseblock3.denselayer16.conv2)
    heatmap = cam.generate_heatmap(img_tensor, top_indices[0])
    orig = image.resize((224, 224)).convert("RGB")
    heatmap_img = np.array(orig) / 255.0
    heatmap_color = cv2.applyColorMap(np.uint8(255 * heatmap),
cv2.COLORMAP_JET)
    overlay = np.uint8(heatmap_color * 0.4 + heatmap_img * 255 * 0.6)
    overlay_img = Image.fromarray(overlay)
    return result, overlay_img

```

```

elif body_part == "Brain MRI":

```

```

img_tensor = preprocess_brain(image)
with torch.no_grad():
    output = brain_model(img_tensor)
    probs = F.softmax(output, dim=1).numpy()[0]
    top_idx = np.argmax(probs)
    result = f"Prediction: {brain_labels[top_idx]} (Confidence:
{probs[top_idx]:.2f})"
    return result, image.resize((224, 224))

elif body_part == "Knee X-ray":
    result = "Demo Prediction: Osteoarthritis (Confidence: 0.82)"
    return result, image.resize((224, 224))

else:
    return "Invalid body part selected.", None

except Exception as e:
    return f"Error: {e}", None

interface = gr.Interface(
    fn=diagnose_with_cam,
    inputs=[
        gr.Image(type="pil", label="Upload Medical Image"),
        gr.Dropdown(label="Select Body Part", choices=["Chest X-ray", "Brain
MRI", "Knee X-ray"], value="Chest X-ray")
    ],
    outputs=["text", "image"],
    title="Multimodal Medical Diagnosis Demo",
    description="Upload an image and select a body part to get predictions.
Chest X-ray supports Grad-CAM. Brain MRI uses classification. Knee X-ray is a
simulated prediction."
)

interface.launch(share=True)

```

## INPUT:

Upload an image and select a body part to get predictions. Chest X-ray supports Grad-CAM. Brain MRI uses classification. Knee X-ray is a simulated prediction.

Upload Medical Image

Drop Image Here

- or -

Click to Upload

📁

🔍

📄

Select Body Part

Clear

Submit

output 0

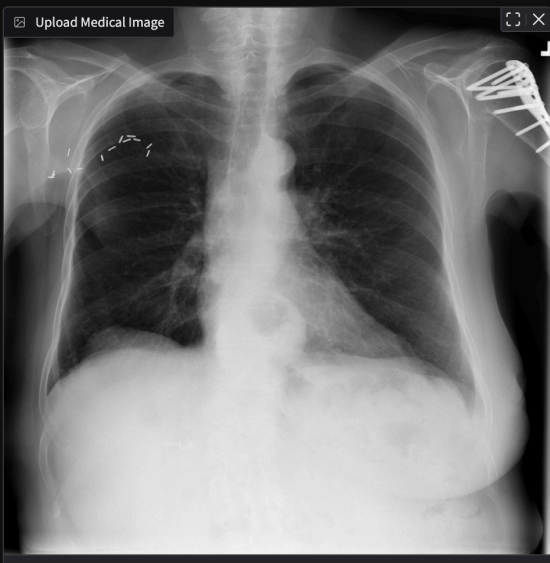
output 1

Flag

## OUTPUT:

Upload an image and select a body part to get predictions. Chest X-ray supports Grad-CAM. Brain MRI uses classification. Knee X-ray is a simulated prediction.

Upload Medical Image



📁

🔍

📄


output 0

Effusion: 0.66

Lung Opacity: 0.66

Enlarged Cardiomeastinum: 0.66

output 1



Flag