import streamlit as st

from PIL import Image

import torch

import cv2

import numpy as np

import torchvision.transforms as transforms

import matplotlib.pyplot as plt

from transformers import ViTForImageClassification, AutoImageProcessor

# Load model and processor

model\_name = "nickmuchi/vit-finetuned-chest-xray-pneumonia"

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

processor = AutoImageProcessor.from\_pretrained(model\_name)

model = ViTForImageClassification.from\_pretrained(model\_name, output\_attentions=True).to(device)

# Function to convert grayscale to RGB

def convert\_to\_rgb(image):

if image.mode != "RGB":

image = image.convert("RGB")

return image

# Function to apply attention visualization with sharpened focus

# Function to apply attention visualization with emphasized critical regions

def apply\_attention\_visualization(image, model, prediction\_idx):

transform = transforms.Compose([

transforms.Resize((224, 224)),

transforms.ToTensor()

])

img\_tensor = transform(image).unsqueeze(0).to(device)

with torch.no\_grad():

outputs = model(img\_tensor, output\_attentions=True)

attentions = outputs.attentions # Extract attention layers

last\_attention = attentions[-1].squeeze(0).mean(dim=0) # Average across heads

# Resize the attention map to match image size

attention\_resized = cv2.resize(last\_attention.mean(dim=0).cpu().numpy(), (image.size[0], image.size[1]))

# Normalize the attention map

attention\_min = attention\_resized.min()

attention\_max = attention\_resized.max()

if attention\_max > attention\_min:

normalized\_attention = (attention\_resized - attention\_min) / (attention\_max - attention\_min)

else:

normalized\_attention = np.zeros\_like(attention\_resized)

image\_np = np.array(image.convert("RGB"), dtype=np.uint8).copy()

if prediction\_idx == 1: # Pneumonia Detected (Emphasized Critical Regions)

# Focus on a very high percentile of attention

threshold = np.percentile(normalized\_attention, 98) # Even higher percentile

# Create a mask for these critical regions

mask = np.uint8(normalized\_attention >= threshold)

# Dilate the mask slightly to make the highlighted areas more visible

kernel = np.ones((5, 5), np.uint8)

dilated\_mask = cv2.dilate(mask, kernel, iterations=1)

# Create a strong red overlay

red\_overlay = np.zeros\_like(image\_np, dtype=np.uint8)

red\_overlay[:, :, 0] = 255 # Red channel

# Overlay the red color on the original image using the dilated mask

masked\_image = cv2.bitwise\_and(red\_overlay, red\_overlay, mask=dilated\_mask)

emphasized\_image = cv2.addWeighted(image\_np, 0.7, masked\_image, 0.3, 0)

return emphasized\_image

else: # No Pneumonia Detected (Subtle Green Overlay)

heatmap = np.uint8(255 \* normalized\_attention)

heatmap\_colored = cv2.applyColorMap(heatmap, cv2.COLORMAP\_VIRIDIS)

overlay = cv2.cvtColor(heatmap\_colored, cv2.COLOR\_RGB2BGR)

blended = cv2.addWeighted(image\_np, 0.7, overlay, 0.3, 0)

return blended

# Function to predict pneumonia

def predict(image):

try:

if not isinstance(image, Image.Image):

raise ValueError("Input is not a PIL Image object")

image\_rgb = image.convert("RGB")

if image\_rgb.mode != "RGB":

raise ValueError(f"Failed to convert image to RGB. Mode is: {image\_rgb.mode}")

if np.array(image\_rgb).shape[-1] != 3:

raise ValueError(f"Image does not have 3 channels. Shape: {np.array(image\_rgb).shape}")

inputs = processor(images=image\_rgb, return\_tensors="pt").to(device)

with torch.no\_grad():

outputs = model(\*\*inputs)

predicted\_class\_idx = torch.argmax(outputs.logits).item()

return predicted\_class\_idx

except Exception as e:

st.error(f"Error in prediction: {e}")

return None

# Streamlit App

def main():

st.title("Pneumonia Detection from Chest X-ray")

st.write("Upload a chest X-ray image to detect pneumonia.")

uploaded\_image = st.file\_uploader("Choose a chest X-ray image...", type=["jpg", "png", "jpeg"])

if uploaded\_image is not None:

image = Image.open(uploaded\_image).convert("RGB") # Force RGB upon loading

st.image(image, caption="Uploaded Image (Converted to RGB)", use\_column\_width=True)

if st.button("Detect Pneumonia"):

prediction\_idx = predict(image)

if prediction\_idx is not None:

if prediction\_idx == 1:

st.error("Pneumonia Detected")

attention\_image = apply\_attention\_visualization(image, model, prediction\_idx)

st.image(attention\_image, caption="Attention (Focused Red Dots - Pneumonia)", use\_column\_width=True)

else:

st.success("No Pneumonia Detected")

attention\_image = apply\_attention\_visualization(image, model, prediction\_idx)

st.image(attention\_image, caption="Attention (Subtle Green Overlay - No Pneumonia)", use\_column\_width=True)

if \_name\_ == "\_main\_":

main()