```
# ========================================
# First, install all required packages
!pip install -q ultralytics opencv-python pydicom


# ========================
import os
import numpy as np
import matplotlib.pyplot as plt
import cv2
import torch
import time
import yaml
import pydicom  # For DICOM image processing
from PIL import Image
from google.colab import files

# Import YOLO library
from ultralytics import YOLO

# Create project directories
base_dir = '/content/lung_cancer_detection'
models_dir = os.path.join(base_dir, 'models')
output_dir = os.path.join(base_dir, 'output')
processed_dir = os.path.join(base_dir, 'processed')

for directory in [base_dir, models_dir, output_dir, processed_dir]:
    os.makedirs(directory, exist_ok=True)

print(f"Directory structure created at {base_dir}")

# Class definitions
CLASSES = ["adenocarcinoma", "large_cell_carcinoma", "normal",
"squamous_cell_carcinoma"]

# Create a data.yaml file for model testing
data_yaml = {
    'names': CLASSES,
    'nc': len(CLASSES)
}

# Save the data.yaml file
with open(os.path.join(base_dir, 'data.yaml'), 'w') as f:
    yaml.dump(data_yaml, f)

print(f"Created data.yaml with classes: {CLASSES}")

# Global variables for cancer pattern explanations with simplified
```

```
definitions
CANCER_PATTERN_EXPLANATIONS = {
    "adenocarcinoma": [
        "Ground glass opacities - hazy areas in the lung that look
like frosted glass on a CT scan",
        "Subsolid nodules - partially solid lumps in the lung tissue",
        "Peripheral lesions - abnormal areas located toward the outer
parts of the lungs",
        "More commonly found in the outer third of the lung - closer
to the chest wall",
        "May present with air bronchograms - air-filled breathing
tubes visible within abnormal tissue"
    ],
    "large_cell_carcinoma": [
        "Large, poorly circumscribed masses - big areas with unclear
or irregular borders",
        "Typically peripheral in location - usually found near the
outer edges of the lungs",
        "Rapid growth and extensive local invasion - spreads quickly
to nearby tissues",
        "Heterogeneous density - appears with varying levels of
brightness on a scan",
        "Less likely to have calcifications - fewer hardened, calcium
deposits"
    ],
    "squamous_cell_carcinoma": [
        "Central masses arising from main or lobar bronchi - tumors
starting in the large air tubes in the center of the lungs",
        "May show cavitation - hollow spaces or holes within the
tumor",
        "Associated with hilar masses - abnormalities near where the
main bronchi enter the lungs",
        "Can present with post-obstructive pneumonia - infection
caused by blocked airways",
        "More likely to show calcifications - hardened, calcium
deposits in the tumor tissue"
    ],
    "normal": [
        "No visible masses, nodules, or suspicious lesions - no
abnormal growths or areas",
        "Normal lung parenchyma - healthy lung tissue shows expected
patterns",
        "Regular bronchial walls and blood vessels tapering normally -
breathing tubes and vessels have uniform thickness",
        "Normal mediastinal contours - the central chest area has
typical appearance",
        "Clear lung fields without consolidation or infiltrates -
lungs appear transparent without dense areas"
    ]
```

```
}

# Additional explanations of medical terms for easier understanding
MEDICAL_TERMS_EXPLAINED = {
    "bronchi": "The large air tubes that carry air from the windpipe
(trachea) into the lungs",
    "main bronchi": "The two large air tubes that branch directly from
the windpipe, one going to each lung",
    "lobar bronchi": "Smaller branches of the main bronchi that lead
to the five lobes (sections) of the lungs",
    "hilar": "The area where the main bronchi, blood vessels, and
nerves enter each lung",
    "parenchyma": "The functional tissue of an organ; in lungs, it's
the tissue involved in gas exchange",
    "mediastinal": "Relating to the mediastinum, the central
compartment of the chest between the lungs",
    "consolidation": "When lung tissue becomes filled with liquid
instead of air",
    "infiltrates": "Substances that are not normally present in the
lungs, such as fluid or cells",
    "nodule": "A small round or oval-shaped growth or lump",
    "cavitation": "The formation of hollow spaces within a solid area
or mass",
    "ground glass opacity": "A hazy, cloudy area on a CT scan that
doesn't completely obscure underlying structures"
}

print("Setup complete!")


# ================================
def read_dicom_image(file_path):
    """
    Read a DICOM image and convert to a format suitable for processing

    Args:
        file_path: Path to the DICOM file

    Returns:
        Image in RGB format
    """
    try:
        # Check if file is a DICOM file
        if file_path.lower().endswith(('.dcm', '.dicom')):
            # Read DICOM file
            ds = pydicom.dcmread(file_path)
            img = ds.pixel_array

            # Normalize to 8-bit range (0-255)
            img = ((img - img.min()) / (img.max() - img.min()) *
```

```
255).astype(np.uint8)

            # Convert to RGB (DICOM images are often grayscale)
            img_rgb = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
            return img_rgb
        else:
            # For non-DICOM images, use OpenCV
            img = cv2.imread(file_path)
            if img is None:
                raise ValueError(f"Could not read image from
{file_path}")
            img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            return img_rgb
    except Exception as e:
        print(f"Error reading image: {e}")
        return None

def is_likely_lung_ct(image, strict_mode=False):
    """
    Basic check to determine if an image is likely a lung CT scan

    Args:
        image: Image to check
        strict_mode: If True, apply stricter validation criteria

    Returns:
        confidence: 0-1 score indicating confidence that this is a
lung CT
        is_valid: Boolean indicating if image passes validation
    """
    try:
        # Convert to grayscale if not already
        if len(image.shape) == 3:
            gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        else:
            gray = image

        # Check image characteristics common in lung CTs:

        # 1. Check for overall brightness/contrast distribution
        # Lung CTs typically have certain brightness levels with dark
lungs and brighter bone/tissue
        hist = cv2.calcHist([gray], [0], None, [256], [0, 256])
        hist_normalized = hist / hist.sum()  # Normalize to get
probability distribution

        # Check if we have significant dark areas (air in lungs shows
as dark)
        dark_ratio = hist_normalized[0:50].sum()
```

```
        # Check if we have significant bright areas (bones, tissue)
        bright_ratio = hist_normalized[150:].sum()

        # 2. Check for circular/elliptical structure (thoracic cavity)
        # Apply threshold to isolate potential lung regions
        _, thresh = cv2.threshold(gray, 80, 255,
cv2.THRESH_BINARY_INV)

        # Find contours
        contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

        # Check if we have significant contours
        has_significant_contours = False
        for contour in contours:
            area = cv2.contourArea(contour)
            if area > (gray.shape[0] * gray.shape[1] * 0.05):  # At
least 5% of image
                has_significant_contours = True
                break

        # 3. Check for bilateral symmetry (lungs are relatively
symmetric)
        # Divide image in half and compare left and right sides
        h, w = gray.shape
        left_half = gray[:, :w//2]
        right_half = gray[:, w//2:]
        right_half_flipped = cv2.flip(right_half, 1)  # Flip
horizontally

        # Resize if needed to make comparable
        if left_half.shape != right_half_flipped.shape:
            min_w = min(left_half.shape[1],
right_half_flipped.shape[1])
            left_half = left_half[:, :min_w]
            right_half_flipped = right_half_flipped[:, :min_w]

        # Calculate symmetry score (lower means more symmetric)
        symmetry_diff = np.abs(left_half.astype(np.float32) -
right_half_flipped.astype(np.float32)).mean()
        symmetry_score = symmetry_diff / 255.0  # Normalize to 0-1
range

        # Calculate confidence score (higher means more likely a lung
CT)
        confidence_factors = {
            'dark_ratio': 1.0 if dark_ratio > 0.15 else dark_ratio /
0.15,
            'bright_ratio': 1.0 if bright_ratio > 0.1 else
bright_ratio / 0.1,
```

```
            'has_contours': 1.0 if has_significant_contours else 0.3,
            'symmetry': 1.0 if symmetry_score < 0.3 else max(0, (0.5 -
symmetry_score) / 0.2)
        }

        # Weight the factors - emphasize contours and dark regions
(lung areas)
        weighted_confidence = (
            0.3 * confidence_factors['dark_ratio'] +
            0.2 * confidence_factors['bright_ratio'] +
            0.3 * confidence_factors['has_contours'] +
            0.2 * confidence_factors['symmetry']
        )

        # Apply different thresholds based on mode
        is_valid = weighted_confidence > (0.7 if strict_mode else 0.5)

        return weighted_confidence, is_valid

    except Exception as e:
        print(f"Error during lung CT validation: {e}")
        return 0.0, False  # If any error occurs, assume it's not a
lung CT

def preprocess_image(image, size=(640, 640), use_edge_detection=True):
    """
    Preprocess an image for the YOLO model

    Args:
        image: Input image (RGB)
        size: Target size for the image
        use_edge_detection: Whether to apply edge detection

    Returns:
        Preprocessed image and edge map
    """
    try:
        # Ensure image is in RGB format
        if len(image.shape) == 2:  # If grayscale
            image = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)

        # Create a copy of the original image
        processed_img = image.copy()

        # Convert to grayscale for edge detection
        gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

        # Apply edge detection if requested
        edge_map = None
        if use_edge_detection:
```

```
            # Apply Gaussian blur to reduce noise
            blurred = cv2.GaussianBlur(gray, (5, 5), 0)

            # Canny edge detection with automatic threshold based on
image median
            median_val = np.median(blurred)
            lower = int(max(0, 0.66 * median_val))
            upper = int(min(255, 1.33 * median_val))
            edge_map = cv2.Canny(blurred, lower, upper)

        # Resize to the target size
        processed_img = cv2.resize(processed_img, size)
        if edge_map is not None:
            edge_map = cv2.resize(edge_map, size)

        return processed_img, edge_map
    except Exception as e:
        print(f"Error preprocessing image: {e}")
        return None, None

def analyze_contours(image, edge_map=None, min_area=100):
    """
    Analyze contours in the image to highlight potential nodules

    Args:
        image: Input image
        edge_map: Pre-computed edge map (if None, calculated from
image)
        min_area: Minimum contour area to consider

    Returns:
        Image with contours, contour features dictionary
    """
    # Create a copy for drawing
    contour_img = image.copy()

    # If no edge map provided, create one
    if edge_map is None:
        gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
        blurred = cv2.GaussianBlur(gray, (5, 5), 0)
        edge_map = cv2.Canny(blurred, 50, 150)

    # Find contours
    contours, hierarchy = cv2.findContours(edge_map,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Filter and analyze contours
    contour_features = []
    for i, contour in enumerate(contours):
        area = cv2.contourArea(contour)
```

```
        if area < min_area:
            continue

        # Calculate contour features
        perimeter = cv2.arcLength(contour, True)
        circularity = 4 * np.pi * area / (perimeter * perimeter) if
perimeter > 0 else 0

        # Get bounding rectangle
        x, y, w, h = cv2.boundingRect(contour)
        aspect_ratio = w / h if h > 0 else 0

        # Draw contour with unique color based on circularity (more
circular = more red)
        color = (int(255 * (1-circularity)), int(255 * (circularity >
0.5)), int(255 * circularity))
        cv2.drawContours(contour_img, [contour], -1, color, 2)

        # Store features
        contour_features.append({
            'id': i,
            'area': area,
            'perimeter': perimeter,
            'circularity': circularity,
            'aspect_ratio': aspect_ratio,
            'boundingRect': (x, y, w, h)
        })

    # Sort contours by area (largest first)
    contour_features.sort(key=lambda x: x['area'], reverse=True)

    return contour_img, contour_features


# ============================================
def load_yolo_model(model_path):
    """Load a YOLO model from a specified path"""
    if not os.path.exists(model_path):
        print(f"Error: Model file not found at {model_path}")
        return None

    try:
        print(f"Loading model from {model_path}...")
        model = YOLO(model_path)

        # Print only essential model information, not the full
architecture
        print(f" Model loaded successfully!")
        print(f" Model classes: {list(model.names.values())}")
```

```
        return model
    except Exception as e:
        print(f"Error loading model: {e}")
        return None

def upload_model():
    """Function to upload a model file"""
    print("Please upload your trained YOLOv12 model file (best.pt):")
    uploaded = files.upload()

    if not uploaded:
        print("No file uploaded")
        return None

    model_file = list(uploaded.keys())[0]

    # Save uploaded model to models directory
    model_path = os.path.join(models_dir, model_file)
    with open(model_path, 'wb') as f:
        f.write(uploaded[model_file])

    print(f"Model saved to {model_path}")
    return model_path

def upload_test_image():
    """Upload a test image"""
    print("Please upload a lung CT scan image to test (DICOM or
standard formats supported):")
    uploaded = files.upload()

    if not uploaded:
        print("No file uploaded")
        return None

    image_file = list(uploaded.keys())[0]

    # Save uploaded image to output directory
    image_path = os.path.join(output_dir, image_file)
    with open(image_path, 'wb') as f:
        f.write(uploaded[image_file])

    print(f"Image saved to {image_path}")
    return image_path

def run_prediction(model, image_path, conf_threshold=0.25):
    """
    Run prediction with edge detection and contour analysis

    Args:
        model: YOLO model
```

```
        image_path: Path to the image file
        conf_threshold: Confidence threshold for detection

    Returns:
        Results, image, edge map, contour features
    """
    try:
        # Start timing
        start_time = time.time()

        # Read the image
        image = read_dicom_image(image_path)
        if image is None:
            return None, None, None, None

        # Check if this is likely a lung CT scan
        if not is_likely_lung_ct(image):
            print("\n ERROR: The uploaded image does not appear to be
a lung CT scan.")
            print("Please upload a proper lung CT scan image for
analysis.")
            return None, None, None, None

        # Preprocess the image
        processed_img, edge_map = preprocess_image(image)
        if processed_img is None:
            return None, None, None, None

        # Save processed image to temporary file for YOLO
        temp_file = os.path.join(processed_dir,
f"temp_{os.path.basename(image_path)}")
        cv2.imwrite(temp_file, cv2.cvtColor(processed_img,
cv2.COLOR_RGB2BGR))

        # Apply contour analysis
        contour_img, contour_features =
analyze_contours(processed_img, edge_map)

        # Run YOLO prediction
        results = model.predict(temp_file, conf=conf_threshold)[0]

        # Remove temporary file
        if os.path.exists(temp_file):
            os.remove(temp_file)

        # Calculate processing time
        processing_time = time.time() - start_time
        print(f"Prediction completed in {processing_time:.2f}
seconds")
```

```
        return results, image, edge_map, contour_features
    except Exception as e:
        print(f"Error during prediction: {e}")
        return None, None, None, None


# ============================================
def visualize_results(results, original_img, edge_map,
contour_features):
    """
    Visualize the results with annotations

    Args:
        results: YOLO prediction results
        original_img: Original image
        edge_map: Edge detection map
        contour_features: Contour analysis features
    """
    # Create contour image
    contour_img, _ = analyze_contours(original_img, edge_map)

    # Create prediction visualization
    pred_img = original_img.copy()

    # If we have valid results
    if results and len(results.boxes) > 0:
        for box in results.boxes:
            x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()
            conf = float(box.conf[0])
            cls_id = int(box.cls[0])
            cls_name = results.names[cls_id]

            # Draw rectangle
            cv2.rectangle(pred_img, (int(x1), int(y1)), (int(x2),
int(y2)), (255, 0, 0), 2)

            # Add label
            label = f"{cls_name} {conf:.2f}"
            cv2.putText(pred_img, label, (int(x1), int(y1)-10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

    # Set up the figure with 1x3 grid
    fig, axes = plt.subplots(1, 3, figsize=(18, 6))

    # Original Image with Prediction
    axes[0].imshow(pred_img)
    axes[0].set_title("Prediction Results")
    axes[0].axis('off')

    # Edge Detection
    axes[1].imshow(edge_map, cmap='gray')
```

```
    axes[1].set_title("Edge Detection")
    axes[1].axis('off')

    # Contour Analysis
    axes[2].imshow(contour_img)
    axes[2].set_title(f"Contour Analysis\n({len(contour_features)}
contours)")
    axes[2].axis('off')

    plt.tight_layout()
    plt.show()

    # Display prediction results
    print("\n==== LUNG CANCER DETECTION RESULTS ====")

    if results and len(results.boxes) > 0:
        print("\nDetection Results:")
        for i, box in enumerate(results.boxes):
            conf = float(box.conf[0])
            cls_id = int(box.cls[0])
            cls_name = results.names[cls_id]

            # Determine if cancer or not
            is_cancer = cls_name != "normal"
            status = "CANCER DETECTED" if is_cancer else "NO CANCER
DETECTED"

            # Print result with formatting
            print(f"\n----------------------------------------")
            print(f"PREDICTION: {cls_name.upper()}")
            print(f"STATUS: {status}")
            print(f"CONFIDENCE: {conf:.2f} ({conf*100:.1f}%)")
            print(f"----------------------------------------")

            # Medical disclaimer
            if is_cancer:
                print("\n⚠ IMPORTANT: This is an AI-assisted detection
result.")
                print("Please consult with a qualified radiologist or
oncologist")
                print("for proper diagnosis and treatment
recommendations.")
                print("Early detection significantly improves
treatment outcomes.")
            else:
                print("\n✓ Normal lung tissue detected. However,
please note that")
                print("this is an AI-assisted screening result and
should be")
                print("verified by a qualified healthcare
```

```
professional.")

    else:
        print("No detections found. The model could not confidently
classify this image.")
        print("Please consult with a healthcare professional for
proper evaluation.")

    # Print explanation of what's happening - with simplified
explanations
    print("\n==== SIMPLIFIED EXPLANATION ====")
    print("This model was trained on a Kaggle lung cancer CT scan
dataset to detect and")
    print("classify different types of lung cancer patterns in medical
images.")

    print("\nWhat you're seeing in the images:")
    print("1. PREDICTION RESULTS: The original scan with any detected
cancer regions highlighted")
    print("2. EDGE DETECTION: Shows the outlines and boundaries of
structures in the lungs")
    print("3. CONTOUR ANALYSIS: Identifies shapes that might be
tumors, color-coded by shape")
    print("   (more red = more circular, which often indicates
nodules)")

    print("\nThe four possible classifications:")
    print("• Adenocarcinoma: The most common type of lung cancer,
typically starting in")
    print("  the outer parts of the lung in the cells that produce
mucus")
    print("• Large Cell Carcinoma: A fast-growing type of lung cancer
with large,")
    print("  abnormal-looking cells")
    print("• Squamous Cell Carcinoma: Cancer that begins in the flat
cells lining the")
    print("  inside of the airways in the lungs")
    print("• Normal: No cancer detected, healthy lung tissue")

    if results and len(results.boxes) > 0:
        # Get the top prediction
        top_box = results.boxes[0]
        top_conf = float(top_box.conf[0])
        top_cls_id = int(top_box.cls[0])
        top_cls_name = results.names[top_cls_id]

        # Explain the specific prediction
        print(f"\nFor this image, the model detected: {top_cls_name}")
        print("Key features typically associated with this
classification include:")
```

```
        for i, explanation in
enumerate(CANCER_PATTERN_EXPLANATIONS[top_cls_name][:3]):
            print(f"  {i+1}. {explanation}")

        # Look for medical terms that might need explanation
        terms_to_explain = []
        for explanation in CANCER_PATTERN_EXPLANATIONS[top_cls_name]:
            for term in MEDICAL_TERMS_EXPLAINED:
                if term.lower() in explanation.lower() and term not in
terms_to_explain:
                    terms_to_explain.append(term)

        # Provide explanations for medical terms
        if terms_to_explain:
            print("\nMedical terms explained:")
            for term in terms_to_explain:
                print(f"• {term}: {MEDICAL_TERMS_EXPLAINED[term]}")

    print("\n==== TECHNICAL DETAILS ====")
    print(f"Contour Analysis: {len(contour_features)} contours
identified")
    if contour_features:
        circular_count = sum(1 for cf in contour_features if
cf['circularity'] > 0.7)
        irregular_count = sum(1 for cf in contour_features if
cf['circularity'] <= 0.7)
        print(f"  - {circular_count} circular contours (potential
nodules)")
        print(f"  - {irregular_count} irregular contours")

        # Explain contour analysis in simple terms
        print("\nWhat this means:")
        print("• Circular shapes (high circularity) often indicate
nodules which may be tumors")
        print("• Irregular shapes can represent various abnormalities
including spreading cancer")
        print("• The number and distribution of these shapes help the
AI make its prediction")

    print("\nRemember: This tool is for screening purposes only and is
not intended")
    print("to replace professional medical advice, diagnosis, or
treatment.")


# ============================================

# Global variable to store the loaded model for reuse
global_model = None
```

```
def upload_and_load_model():
    """
    Function to upload and load a lung cancer detection model.
    Stores the model in a global variable for reuse.

    Returns:
        True if model was successfully loaded, False otherwise
    """
    global global_model

    print("\n==== MODEL UPLOAD AND LOADING ====")
    print("Please upload your trained YOLOv12 model file (best.pt)")

    # Upload the model
    model_path = upload_model()
    if not model_path:
        print("No model file was uploaded.")
        return False

    # Load the model
    model = load_yolo_model(model_path)
    if not model:
        print("Error loading the model.")
        return False

    # Store in global variable for reuse
    global_model = model
    print("\n Model successfully loaded and ready for testing
images!")
    return True

def test_image():
    """
    Function to test a lung cancer image using the loaded model.

    Returns:
        True if analysis was completed, False otherwise
    """
    global global_model

    # Check if a model is loaded
    if global_model is None:
        print("\n No model is currently loaded. Please upload and
load a model first.")
        return False

    print("\n==== IMAGE TESTING ====")
    print("Please upload a lung CT scan image to test")

    # Upload test image
```

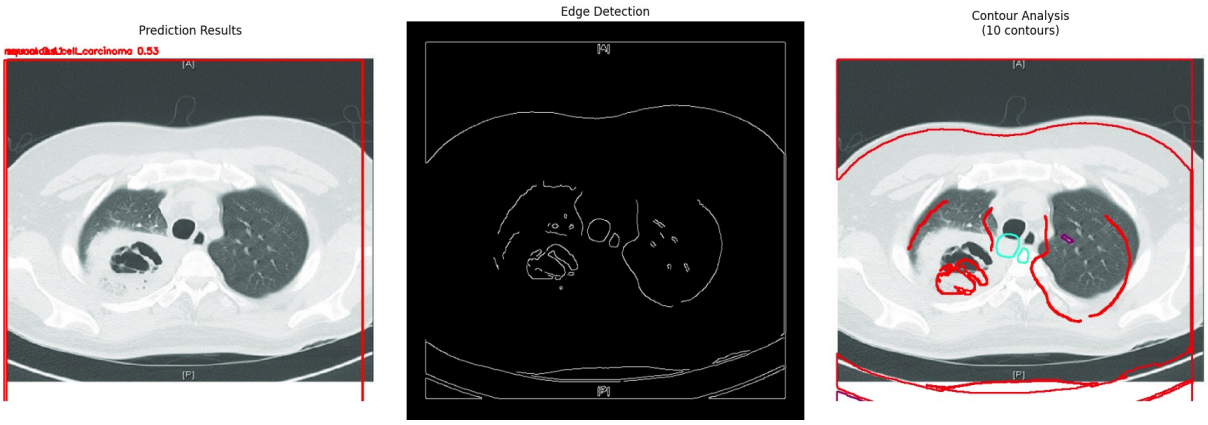
```
    image_path = upload_test_image()
    if not image_path:
        print("No image was uploaded.")
        return False

    # Run prediction
    print("\n--- Running lung cancer detection ---")
    results, original_img, edge_map, contour_features =
run_prediction(global_model, image_path)
    if results is None:
        print("Error processing the image. Please try again with a
different image.")
        return False

    # Visualize and explain results
    visualize_results(results, original_img, edge_map,
contour_features)

    print("\n==== Analysis complete ====")
    print("Remember that this is an AI-assisted screening tool,")
    print("and all results should be confirmed by qualified medical
professionals.")
    return True

def run_batch_test():
    """
    Function to test multiple images with the same model
    """
    global global_model

    # Check if a model is loaded
    if global_model is None:
        print("\n No model is currently loaded. Please upload and
load a model first.")
        upload_status = upload_and_load_model()
        if not upload_status:
            return

    # Process multiple images with the loaded model
    continue_testing = True
    image_count = 0

    while continue_testing:
        image_count += 1
        print(f"\n--- Testing image #{image_count} ---")

        test_status = test_image()
        if not test_status:
            break
```

```
        # Ask if user wants to test another image
        try:
            user_input = input("\nDo you want to test another image
with the same model? (y/n): ").strip().lower()
            continue_testing = user_input.startswith('y')
        except:
            continue_testing = False

    print("\nBatch testing session ended.")

def lung_cancer_detection_menu():
    """
    Main menu function for lung cancer detection
    """
    global global_model

    print("\
n==========================================================================
=")
    print("==== LUNG CANCER DETECTION USING YOLOv12 - INTERACTIVE
INTERFACE ====")

print("==========================================================================
=======")
    print("This notebook allows you to test a trained YOLOv12 model")
    print("for lung cancer detection on medical images.")
    print("\nNote: This tool is for research and educational purposes
only")
    print("and should not be used for clinical diagnosis without
medical supervision.")
    print("\nThis model was trained on a Kaggle lung cancer CT scan
dataset.")

print("==========================================================================
=======")

    while True:
        print("\nMAIN MENU:")
        print("1. Upload and load a model")
        print("2. Test a single image")
        print("3. Run batch testing (multiple images)")
        print("4. Exit")

        model_status = "□ LOADED" if global_model is not None else "□
NOT LOADED"
        print(f"\nCurrent model status: {model_status}")

        choice = input("\nEnter your choice (1-4): ")

        if choice == '1':
```

```
            upload_and_load_model()
        elif choice == '2':
            test_image()
        elif choice == '3':
            run_batch_test()
        elif choice == '4':
            print("\nExiting lung cancer detection system. Thank you
for using our tool!")
            break
        else:
            print("\nInvalid choice. Please enter a number between 1
and 4.")


━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 17.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 50.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 101.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 82.4 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 43.7 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.2 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 6.1 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 6.6 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 7.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 39.1 MB/s eta
0:00:00
/quickstart/#ultralytics-settings.
Directory structure created at /content/lung_cancer_detection
Created data.yaml with classes: ['adenocarcinoma',
'large_cell_carcinoma', 'normal', 'squamous_cell_carcinoma']
Setup complete!

# Run the main menu
lung_cancer_detection_menu()


================================================================
==== LUNG CANCER DETECTION USING YOLOv12 - INTERACTIVE INTERFACE ====
```

```
================================================================================
This notebook allows you to test a trained YOLOv12 model
for lung cancer detection on medical images.

Note: This tool is for research and educational purposes only
and should not be used for clinical diagnosis without medical
supervision.

This model was trained on a Kaggle lung cancer CT scan dataset.
================================================================================

MAIN MENU:
1. Upload and load a model
2. Test a single image
3. Run batch testing (multiple images)
4. Exit

Current model status:  NOT LOADED

==== MODEL UPLOAD AND LOADING ====
Please upload your trained YOLOv12 model file (best.pt)
Please upload your trained YOLOv12 model file (best.pt):

<IPython.core.display.HTML object>

Saving best.pt to best (1).pt
Model saved to /content/lung_cancer_detection/models/best (1).pt
Loading model from /content/lung_cancer_detection/models/best
(1).pt...
 Model loaded successfully!
 Model classes: ['adenocarcinoma', 'large_cell_carcinoma', 'normal',
'squamous_cell_carcinoma']

 Model successfully loaded and ready for testing images!

MAIN MENU:
1. Upload and load a model
2. Test a single image
3. Run batch testing (multiple images)
4. Exit

Current model status:  LOADED

--- Testing image #1 ---

==== IMAGE TESTING ====
Please upload a lung CT scan image to test
Please upload a lung CT scan image to test (DICOM or standard formats
supported):

<IPython.core.display.HTML object>
```

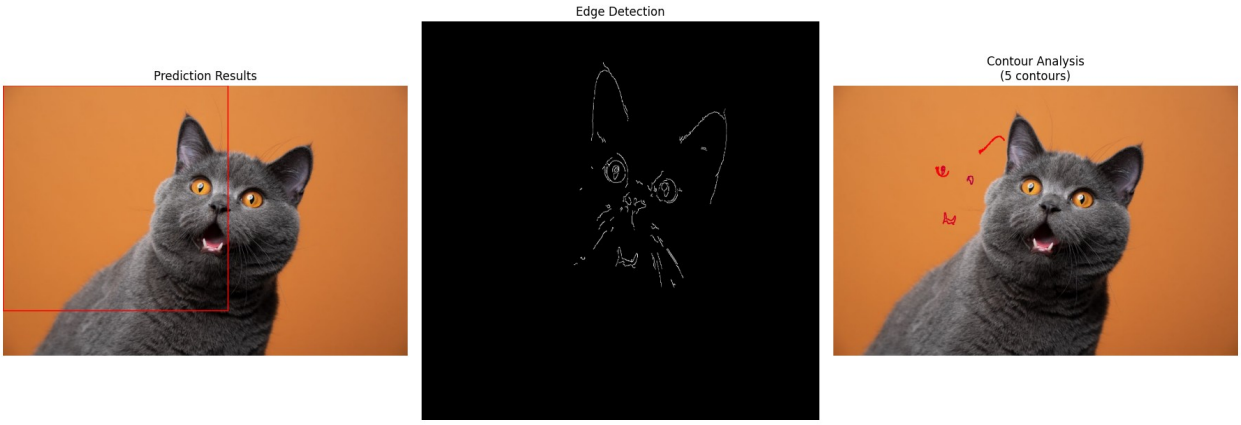
```
Saving Squamous Cell Carcinoma 1.jpg to Squamous Cell Carcinoma 1.jpg
Saving Adenocarcinoma 1.jpg to Adenocarcinoma 1.jpg
Saving Squamous Cell Carcinoma 3.jpg to Squamous Cell Carcinoma 3.jpg
Saving Adenocarcinoma 4.png to Adenocarcinoma 4.png
Saving TRAIN LC 3.png to TRAIN LC 3.png
Saving pusa.jpg to pusa.jpg
Saving TRAIN AD 1.png to TRAIN AD 1.png
Saving TRAIN NORMAL 3.png to TRAIN NORMAL 3.png
Image saved to /content/lung_cancer_detection/output/Squamous Cell
Carcinoma 1.jpg

--- Running lung cancer detection ---

image 1/1 /content/lung_cancer_detection/processed/temp_Squamous Cell
Carcinoma 1.jpg: 640x640 1 normal, 1 squamous_cell_carcinoma, 648.9ms
Speed: 10.7ms preprocess, 648.9ms inference, 34.8ms postprocess per
image at shape (1, 3, 640, 640)
Prediction completed in 9.26 seconds
```




```
==== LUNG CANCER DETECTION RESULTS ====

Detection Results:

----------------------------------------
PREDICTION: SQUAMOUS_CELL_CARCINOMA
STATUS: CANCER DETECTED
CONFIDENCE: 0.53 (52.7%)
----------------------------------------

⚠ IMPORTANT: This is an AI-assisted detection result.
Please consult with a qualified radiologist or oncologist
for proper diagnosis and treatment recommendations.
Early detection significantly improves treatment outcomes.

----------------------------------------
```

```
PREDICTION: NORMAL
STATUS: NO CANCER DETECTED
CONFIDENCE: 0.41 (41.3%)
--------------------------------------

✓ Normal lung tissue detected. However, please note that
this is an AI-assisted screening result and should be
verified by a qualified healthcare professional.

==== SIMPLIFIED EXPLANATION ====
This model was trained on a Kaggle lung cancer CT scan dataset to
detect and
classify different types of lung cancer patterns in medical images.

What you're seeing in the images:
1. PREDICTION RESULTS: The original scan with any detected cancer
regions highlighted
2. EDGE DETECTION: Shows the outlines and boundaries of structures in
the lungs
3. CONTOUR ANALYSIS: Identifies shapes that might be tumors, color-
coded by shape
   (more red = more circular, which often indicates nodules)

The four possible classifications:
• Adenocarcinoma: The most common type of lung cancer, typically
starting in
  the outer parts of the lung in the cells that produce mucus
• Large Cell Carcinoma: A fast-growing type of lung cancer with large,
  abnormal-looking cells
• Squamous Cell Carcinoma: Cancer that begins in the flat cells lining
the
  inside of the airways in the lungs
• Normal: No cancer detected, healthy lung tissue

For this image, the model detected: squamous_cell_carcinoma
Key features typically associated with this classification include:
  1. Central masses arising from main or lobar bronchi - tumors
starting in the large air tubes in the center of the lungs
  2. May show cavitation - hollow spaces or holes within the tumor
  3. Associated with hilar masses - abnormalities near where the main
bronchi enter the lungs

Medical terms explained:
• bronchi: The large air tubes that carry air from the windpipe
(trachea) into the lungs
• lobar bronchi: Smaller branches of the main bronchi that lead to the
five lobes (sections) of the lungs
• cavitation: The formation of hollow spaces within a solid area or
mass
• main bronchi: The two large air tubes that branch directly from the
```

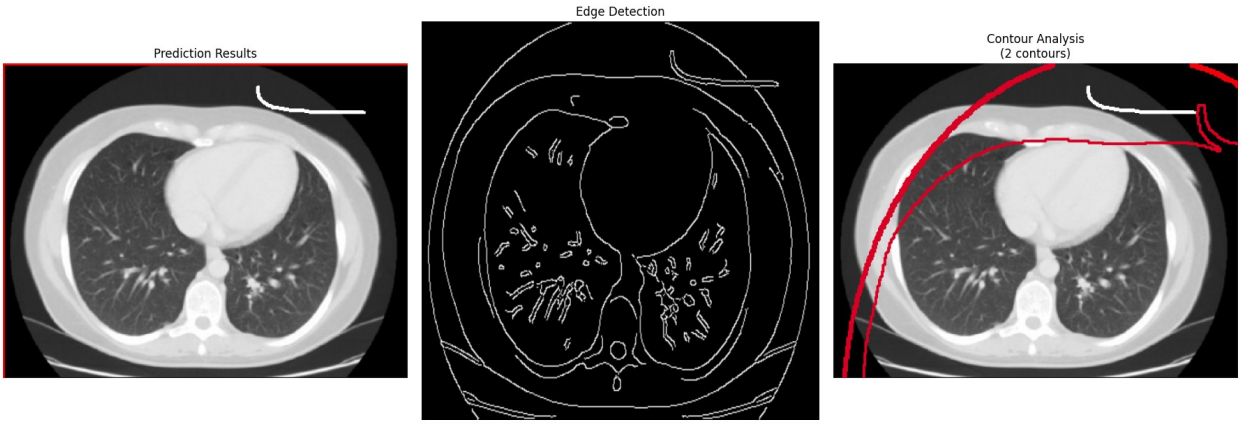
```
windpipe, one going to each lung
• hilar: The area where the main bronchi, blood vessels, and nerves
enter each lung

==== TECHNICAL DETAILS ====
Contour Analysis: 10 contours identified
  - 2 circular contours (potential nodules)
  - 8 irregular contours

What this means:
• Circular shapes (high circularity) often indicate nodules which may
be tumors
• Irregular shapes can represent various abnormalities including
spreading cancer
• The number and distribution of these shapes help the AI make its
prediction

Remember: This tool is for screening purposes only and is not intended
to replace professional medical advice, diagnosis, or treatment.

==== Analysis complete ====
Remember that this is an AI-assisted screening tool,
and all results should be confirmed by qualified medical
professionals.

--- Testing image #2 ---

==== IMAGE TESTING ====
Please upload a lung CT scan image to test
Please upload a lung CT scan image to test (DICOM or standard formats
supported):

<IPython.core.display.HTML object>

Saving pusa.jpg to pusa (1).jpg
Image saved to /content/lung_cancer_detection/output/pusa (1).jpg

--- Running lung cancer detection ---

image 1/1 /content/lung_cancer_detection/processed/temp_pusa (1).jpg:
640x640 1 normal, 271.6ms
Speed: 9.5ms preprocess, 271.6ms inference, 1.2ms postprocess per
image at shape (1, 3, 640, 640)
Prediction completed in 0.32 seconds
```




```
==== LUNG CANCER DETECTION RESULTS ====

Detection Results:

---------------------------------------
PREDICTION: NORMAL
STATUS: NO CANCER DETECTED
CONFIDENCE: 0.84 (83.9%)
---------------------------------------

✓ Normal lung tissue detected. However, please note that
this is an AI-assisted screening result and should be
verified by a qualified healthcare professional.

==== SIMPLIFIED EXPLANATION ====
This model was trained on a Kaggle lung cancer CT scan dataset to
detect and
classify different types of lung cancer patterns in medical images.

What you're seeing in the images:
1. PREDICTION RESULTS: The original scan with any detected cancer
regions highlighted
2. EDGE DETECTION: Shows the outlines and boundaries of structures in
the lungs
3. CONTOUR ANALYSIS: Identifies shapes that might be tumors, color-
coded by shape
   (more red = more circular, which often indicates nodules)

The four possible classifications:
• Adenocarcinoma: The most common type of lung cancer, typically
starting in
  the outer parts of the lung in the cells that produce mucus
• Large Cell Carcinoma: A fast-growing type of lung cancer with large,
  abnormal-looking cells
• Squamous Cell Carcinoma: Cancer that begins in the flat cells lining
the
```

```
  inside of the airways in the lungs
• Normal: No cancer detected, healthy lung tissue

For this image, the model detected: normal
Key features typically associated with this classification include:
  1. No visible masses, nodules, or suspicious lesions - no abnormal
growths or areas
  2. Normal lung parenchyma - healthy lung tissue shows expected
patterns
  3. Regular bronchial walls and blood vessels tapering normally -
breathing tubes and vessels have uniform thickness

Medical terms explained:
• nodule: A small round or oval-shaped growth or lump
• parenchyma: The functional tissue of an organ; in lungs, it's the
tissue involved in gas exchange
• bronchi: The large air tubes that carry air from the windpipe
(trachea) into the lungs
• mediastinal: Relating to the mediastinum, the central compartment of
the chest between the lungs
• consolidation: When lung tissue becomes filled with liquid instead
of air
• infiltrates: Substances that are not normally present in the lungs,
such as fluid or cells

==== TECHNICAL DETAILS ====
Contour Analysis: 5 contours identified
  - 0 circular contours (potential nodules)
  - 5 irregular contours

What this means:
• Circular shapes (high circularity) often indicate nodules which may
be tumors
• Irregular shapes can represent various abnormalities including
spreading cancer
• The number and distribution of these shapes help the AI make its
prediction

Remember: This tool is for screening purposes only and is not intended
to replace professional medical advice, diagnosis, or treatment.

==== Analysis complete ====
Remember that this is an AI-assisted screening tool,
and all results should be confirmed by qualified medical
professionals.

--- Testing image #3 ---

==== IMAGE TESTING ====
Please upload a lung CT scan image to test
```

```
Please upload a lung CT scan image to test (DICOM or standard formats
supported):

<IPython.core.display.HTML object>

Saving TRAIN NORMAL 2.png to TRAIN NORMAL 2.png
Image saved to /content/lung_cancer_detection/output/TRAIN NORMAL
2.png

--- Running lung cancer detection ---

image 1/1 /content/lung_cancer_detection/processed/temp_TRAIN NORMAL
2.png: 640x640 1 normal, 260.1ms
Speed: 4.8ms preprocess, 260.1ms inference, 1.2ms postprocess per
image at shape (1, 3, 640, 640)
Prediction completed in 0.31 seconds
```




```


==== LUNG CANCER DETECTION RESULTS ====

Detection Results:

--------------------------------------
PREDICTION: NORMAL
STATUS: NO CANCER DETECTED
CONFIDENCE: 0.99 (99.3%)
--------------------------------------

✓ Normal lung tissue detected. However, please note that
this is an AI-assisted screening result and should be
verified by a qualified healthcare professional.

==== SIMPLIFIED EXPLANATION ====
This model was trained on a Kaggle lung cancer CT scan dataset to
detect and
classify different types of lung cancer patterns in medical images.
```

What you're seeing in the images:
1. PREDICTION RESULTS: The original scan with any detected cancer regions highlighted
2. EDGE DETECTION: Shows the outlines and boundaries of structures in the lungs
3. CONTOUR ANALYSIS: Identifies shapes that might be tumors, color-coded by shape
   (more red = more circular, which often indicates nodules)

The four possible classifications:
• Adenocarcinoma: The most common type of lung cancer, typically starting in
  the outer parts of the lung in the cells that produce mucus
• Large Cell Carcinoma: A fast-growing type of lung cancer with large,
  abnormal-looking cells
• Squamous Cell Carcinoma: Cancer that begins in the flat cells lining the
  inside of the airways in the lungs
• Normal: No cancer detected, healthy lung tissue

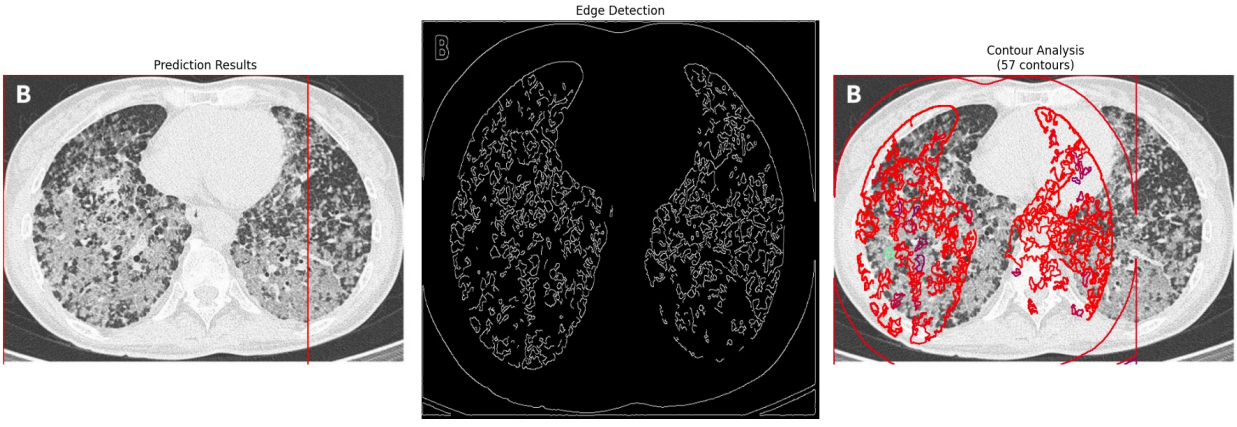
For this image, the model detected: normal
Key features typically associated with this classification include:
  1. No visible masses, nodules, or suspicious lesions - no abnormal growths or areas
  2. Normal lung parenchyma - healthy lung tissue shows expected patterns
  3. Regular bronchial walls and blood vessels tapering normally - breathing tubes and vessels have uniform thickness

Medical terms explained:
• nodule: A small round or oval-shaped growth or lump
• parenchyma: The functional tissue of an organ; in lungs, it's the tissue involved in gas exchange
• bronchi: The large air tubes that carry air from the windpipe (trachea) into the lungs
• mediastinal: Relating to the mediastinum, the central compartment of the chest between the lungs
• consolidation: When lung tissue becomes filled with liquid instead of air
• infiltrates: Substances that are not normally present in the lungs, such as fluid or cells

==== TECHNICAL DETAILS ====
Contour Analysis: 2 contours identified
  - 0 circular contours (potential nodules)
  - 2 irregular contours

What this means:
• Circular shapes (high circularity) often indicate nodules which may be tumors

```
• Irregular shapes can represent various abnormalities including spreading cancer
• The number and distribution of these shapes help the AI make its prediction

Remember: This tool is for screening purposes only and is not intended to replace professional medical advice, diagnosis, or treatment.

==== Analysis complete ====
Remember that this is an AI-assisted screening tool,
and all results should be confirmed by qualified medical professionals.

--- Testing image #4 ---

==== IMAGE TESTING ====
Please upload a lung CT scan image to test
Please upload a lung CT scan image to test (DICOM or standard formats supported):

<IPython.core.display.HTML object>

Saving Adenocarcinoma 4.png to Adenocarcinoma 4 (1).png
Image saved to /content/lung_cancer_detection/output/Adenocarcinoma 4 (1).png

--- Running lung cancer detection ---

image 1/1 /content/lung_cancer_detection/processed/temp_Adenocarcinoma 4 (1).png: 640x640 1 normal, 335.0ms
Speed: 8.0ms preprocess, 335.0ms inference, 1.6ms postprocess per image at shape (1, 3, 640, 640)
Prediction completed in 0.47 seconds
```




```
==== LUNG CANCER DETECTION RESULTS ====
```

```
Detection Results:

--------------------------------------
PREDICTION: NORMAL
STATUS: NO CANCER DETECTED
CONFIDENCE: 0.98 (98.0%)
--------------------------------------

✓ Normal lung tissue detected. However, please note that
this is an AI-assisted screening result and should be
verified by a qualified healthcare professional.

==== SIMPLIFIED EXPLANATION ====
This model was trained on a Kaggle lung cancer CT scan dataset to
detect and
classify different types of lung cancer patterns in medical images.

What you're seeing in the images:
1. PREDICTION RESULTS: The original scan with any detected cancer
regions highlighted
2. EDGE DETECTION: Shows the outlines and boundaries of structures in
the lungs
3. CONTOUR ANALYSIS: Identifies shapes that might be tumors, color-
coded by shape
   (more red = more circular, which often indicates nodules)

The four possible classifications:
• Adenocarcinoma: The most common type of lung cancer, typically
starting in
  the outer parts of the lung in the cells that produce mucus
• Large Cell Carcinoma: A fast-growing type of lung cancer with large,
  abnormal-looking cells
• Squamous Cell Carcinoma: Cancer that begins in the flat cells lining
the
  inside of the airways in the lungs
• Normal: No cancer detected, healthy lung tissue

For this image, the model detected: normal
Key features typically associated with this classification include:
  1. No visible masses, nodules, or suspicious lesions - no abnormal
growths or areas
  2. Normal lung parenchyma - healthy lung tissue shows expected
patterns
  3. Regular bronchial walls and blood vessels tapering normally -
breathing tubes and vessels have uniform thickness

Medical terms explained:
• nodule: A small round or oval-shaped growth or lump
• parenchyma: The functional tissue of an organ; in lungs, it's the
tissue involved in gas exchange
```

```
• bronchi: The large air tubes that carry air from the windpipe
(trachea) into the lungs
• mediastinal: Relating to the mediastinum, the central compartment of
the chest between the lungs
• consolidation: When lung tissue becomes filled with liquid instead
of air
• infiltrates: Substances that are not normally present in the lungs,
such as fluid or cells

==== TECHNICAL DETAILS ====
Contour Analysis: 57 contours identified
  - 0 circular contours (potential nodules)
  - 57 irregular contours

What this means:
• Circular shapes (high circularity) often indicate nodules which may
be tumors
• Irregular shapes can represent various abnormalities including
spreading cancer
• The number and distribution of these shapes help the AI make its
prediction

Remember: This tool is for screening purposes only and is not intended
to replace professional medical advice, diagnosis, or treatment.

==== Analysis complete ====
Remember that this is an AI-assisted screening tool,
and all results should be confirmed by qualified medical
professionals.

--- Testing image #5 ---

==== IMAGE TESTING ====
Please upload a lung CT scan image to test
Please upload a lung CT scan image to test (DICOM or standard formats
supported):

<IPython.core.display.HTML object>

Saving Large Cell Carcinoma 4.jpg to Large Cell Carcinoma 4.jpg
Image saved to /content/lung_cancer_detection/output/Large Cell
Carcinoma 4.jpg

--- Running lung cancer detection ---

image 1/1 /content/lung_cancer_detection/processed/temp_Large Cell
Carcinoma 4.jpg: 640x640 1 squamous_cell_carcinoma, 291.2ms
Speed: 8.3ms preprocess, 291.2ms inference, 1.4ms postprocess per
image at shape (1, 3, 640, 640)
Prediction completed in 0.32 seconds
```


```
==== LUNG CANCER DETECTION RESULTS ====

Detection Results:

----------------------------------------
PREDICTION: SQUAMOUS_CELL_CARCINOMA
STATUS: CANCER DETECTED
CONFIDENCE: 0.97 (97.2%)
----------------------------------------

⚠ IMPORTANT: This is an AI-assisted detection result.
Please consult with a qualified radiologist or oncologist
for proper diagnosis and treatment recommendations.
Early detection significantly improves treatment outcomes.

==== SIMPLIFIED EXPLANATION ====
This model was trained on a Kaggle lung cancer CT scan dataset to
detect and
classify different types of lung cancer patterns in medical images.

What you're seeing in the images:
1. PREDICTION RESULTS: The original scan with any detected cancer
regions highlighted
2. EDGE DETECTION: Shows the outlines and boundaries of structures in
the lungs
3. CONTOUR ANALYSIS: Identifies shapes that might be tumors, color-
coded by shape
   (more red = more circular, which often indicates nodules)

The four possible classifications:
• Adenocarcinoma: The most common type of lung cancer, typically
starting in
  the outer parts of the lung in the cells that produce mucus
• Large Cell Carcinoma: A fast-growing type of lung cancer with large,
  abnormal-looking cells
• Squamous Cell Carcinoma: Cancer that begins in the flat cells lining
```

```
the
  inside of the airways in the lungs
• Normal: No cancer detected, healthy lung tissue

For this image, the model detected: squamous_cell_carcinoma
Key features typically associated with this classification include:
  1. Central masses arising from main or lobar bronchi - tumors
starting in the large air tubes in the center of the lungs
  2. May show cavitation - hollow spaces or holes within the tumor
  3. Associated with hilar masses - abnormalities near where the main
bronchi enter the lungs

Medical terms explained:
• bronchi: The large air tubes that carry air from the windpipe
(trachea) into the lungs
• lobar bronchi: Smaller branches of the main bronchi that lead to the
five lobes (sections) of the lungs
• cavitation: The formation of hollow spaces within a solid area or
mass
• main bronchi: The two large air tubes that branch directly from the
windpipe, one going to each lung
• hilar: The area where the main bronchi, blood vessels, and nerves
enter each lung

==== TECHNICAL DETAILS ====
Contour Analysis: 18 contours identified
  - 2 circular contours (potential nodules)
  - 16 irregular contours

What this means:
• Circular shapes (high circularity) often indicate nodules which may
be tumors
• Irregular shapes can represent various abnormalities including
spreading cancer
• The number and distribution of these shapes help the AI make its
prediction

Remember: This tool is for screening purposes only and is not intended
to replace professional medical advice, diagnosis, or treatment.

==== Analysis complete ====
Remember that this is an AI-assisted screening tool,
and all results should be confirmed by qualified medical
professionals.

Batch testing session ended.

MAIN MENU:
1. Upload and load a model
2. Test a single image
```

```
3. Run batch testing (multiple images)
4. Exit

Current model status:  LOADED
```