# LungXAI Project Review Guide

## Complete Documentation for Project Defense & Viva

---

# TABLE OF CONTENTS



---

# 1. PROJECT OVERVIEW

## 1.1 Project Title

"Explainable AI for Multi-Class Lung Cancer Classification Using Deep Learning and RAG-Based Knowledge Retrieval"

## 1.2 One-Line Summary

A deep learning system that not only classifies lung CT scan images into cancer types but also provides visual explanations (WHERE the model looks) and textual explanations (WHY those regions matter) using Grad-CAM and RAG.

## 1.3 Key Innovation

| Traditional AI | Our System |
|---|---|
| Only predicts class | Predicts + Explains |
| Black-box | Transparent |
| No medical context | Provides medical knowledge |
| Single output | Visual + Textual explanation |

## 1.4 Classes Classified (5 Classes)

| Class | Description | Typical Location |
|---|---|---|
| Adenocarcinoma | Most common lung cancer (~40%) | Peripheral/Outer regions |
| Squamous Cell Carcinoma | Second most common (~30%) | Central/Hilar regions |
| Large Cell Carcinoma | Fast-growing, aggressive | Peripheral, larger masses |
| Benign Cases | Non-cancerous lesions | Variable |
| Normal Cases | Healthy lung tissue | Clear patterns |

---

# 2. PROBLEM STATEMENT & MOTIVATION

## 2.1 The Problem

- Lung cancer is the leading cause of cancer deaths worldwide
- Early detection is critical - 5-year survival rate is 56% if detected early vs 5% if detected late
- Radiologist shortage - Increasing CT scan volumes with limited experts
- AI trust issue - Doctors don't trust black-box AI predictions

## 2.2 Why Explainability Matters

```
[X] Without Explainability:
   AI says: "Adenocarcinoma (92%)"
   Doctor thinks: "But why? I can't trust this."

[OK] With Explainability:
   AI says: "Adenocarcinoma (92%)"
   + Shows: Heatmap highlighting peripheral region
   + Explains: "Adenocarcinoma typically presents in peripheral
                regions. Model focused on upper-right peripheral
                area with ground-glass opacity pattern."
   Doctor thinks: "This aligns with clinical knowledge. I can verify."
```

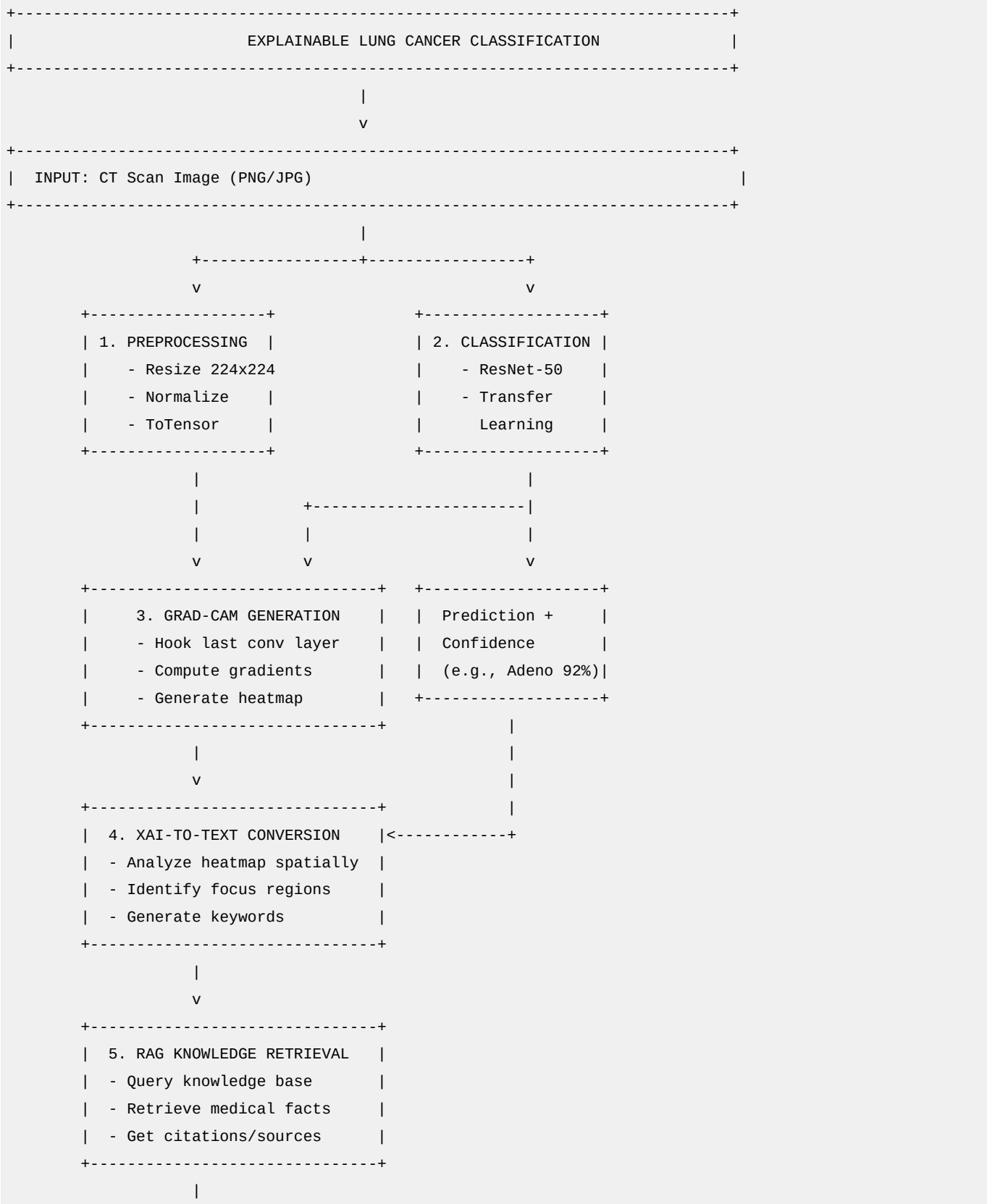
## 2.3 Project Objectives

- Classify lung CT images into 5 categories with high accuracy
- Visualize model attention using Grad-CAM
- Explain predictions using medical knowledge (RAG approach)
- Bridge the gap between AI predictions and clinical understanding

---

# 3. COMPLETE ARCHITECTURE

## 3.1 High-Level Pipeline



```
                    v
+-------------------------------------------------------------------------+
|  OUTPUT:                                                                |
|  +-----------------+  +-----------------+  +----------------------------+  |
|  | Original Image  |  | Grad-CAM Heatmap|  | Textual Explanation        |  |
|  |                 |  |  (Where model   |  | - Visual evidence          |  |
|  |                 |  |   focused)      |  | - Medical context          |  |
|  |                 |  |                 |  | - Cited sources            |  |
|  +-----------------+  +-----------------+  +----------------------------+  |
+-------------------------------------------------------------------------+
```

## 3.2 Data Flow Diagram

```
CT Image -> Preprocessing -> Model Forward Pass -> Prediction
                                v
                          Feature Maps (layer4)
                                v
                          Backward Pass -> Gradients
                                v
                          Grad-CAM Heatmap
                                v
                          Spatial Analysis -> Keywords
                                v
                          Knowledge Retrieval -> Medical Facts
                                v
                          Combined Explanation
```

---

# 4. TECHNOLOGY STACK - WHAT, WHERE & WHY

## 4.1 Core Framework

| Technology | What Is It | Where Used | Why Used |
|---|---|---|---|
| Python 3.10+ | Programming language | Entire project | Industry standard for ML, e... |
| PyTorch 2.0+ | Deep learning framework | Model, training, inference | Academic flexibility, dynam... |
| torchvision | Vision library | Pretrained models, transforms | Official PyTorch extension ... |

## 4.2 Model Architecture

| Technology | What Is It | Where Used | Why Used |
|---|---|---|---|
| ResNet-50 | 50-layer residual network | classifier.py | Proven performance, residua... |
| Transfer Learning | Reusing pretrained weights | Model initialization | ImageNet pretrained weights... |
| Dropout (0.5) | Regularization technique | Final FC layer | Prevents overfitting, impro... |

### Why ResNet-50 Specifically?

```
ResNet-50 Benefits:
|-- [OK] 50 layers deep - captures complex patterns
|-- [OK] Residual connections - solves vanishing gradient
|-- [OK] 2048 features before FC - rich representations
|-- [OK] Well-documented - easy to explain in viva
|-- [OK] Grad-CAM compatible - layer4 gives clear heatmaps
+-- [OK] ImageNet pretrained - 1000-class general features

Alternative Considered:
|-- VGG-16: Simpler but fewer features, no residuals
|-- EfficientNet: More efficient but harder to explain
+-- ViT: Newer but attention mechanism different from Grad-CAM
```

## 4.3 Explainability Stack

| Technology | What Is It | Where Used | Why Used |
|---|---|---|---|
| Grad-CAM | Gradient-weighted Class Act... | gradcam.py | Visual explanation, shows i... |
| OpenCV | Computer vision library | visualize.py | Heatmap overlay, colormap a... |
| Matplotlib | Plotting library | Visualization | Creating publication-qualit... |

## 4.4 RAG (Retrieval-Augmented Generation) Stack

| Technology | What Is It | Where Used | Why Used |
|---|---|---|---|
| Custom Knowledge Base | JSON-structured medical facts | knowledge_base.py | Verified content, no halluc... |
| Keyword Matching | Simple retrieval method | Knowledge retrieval | Transparent, explainable, s... |
| XAI-to-Text Converter | Heatmap analysis | xai_to_text.py | Novel contribution - bridge... |

## 4.5 Data Processing

| Technology | What Is It | Where Used | Why Used |
|---|---|---|---|
| PIL (Pillow) | Image loading | dataset.py | Standard Python imaging lib... |
| NumPy | Numerical operations | Throughout | Array operations, statistics |
| scikit-learn | ML utilities | metrics.py, data splitting | Train/test split, evaluatio... |
| tqdm | Progress bars | Training loops | Visual progress tracking |

## 4.6 Image Preprocessing Details

| Transform | Parameter | Why |
|---|---|---|
| Resize | 224x224 | Standard input size for Ima... |
| Normalize | mean=[0.485,0.456,0.406], s... | ImageNet statistics - requi... |
| RandomHorizontalFlip | p=0.5 | Data augmentation - lungs a... |
| RandomRotation | +/-15 deg | Data augmentation - handles... |
| ColorJitter | brightness/contrast=0.1 | Data augmentation - handles... |

### Why NOT Vertical Flip?

> *Anatomically, lungs have top-bottom orientation that matters clinically. Flipping vertically would create unrealistic images.*

---

# 5. MODULE-BY-MODULE BREAKDOWN

## 5.1 src/data/ - Data Handling

### dataset.py - Custom PyTorch Dataset

```
class LungCancerDataset(Dataset):
    """
    WHAT: Custom dataset class for loading CT images
    WHY: Full control over data loading, can add metadata later
    HOW:
        - Scans class folders
        - Loads images with PIL
        - Applies transforms
        - Returns (image_tensor, label) pairs
    """
```

Key Methods:

| Method | Purpose |
|---|---|
| __init__ | Initialize paths, detect cl... |
| __len__ | Return total number of images |
| __getitem__ | Load single image, apply tr... |

### transforms.py - Image Preprocessing

```
# Training: Augmentation + Normalization
get_train_transforms() -> Resize -> Flip -> Rotate -> ColorJitter -> ToTensor -> Normalize

# Validation/Test: Only Normalization (no augmentation)
get_val_transforms() -> Resize -> ToTensor -> Normalize
```

Why Separate Transforms?

> *Training needs augmentation to increase data variety. Validation/Test needs deterministic, consistent preprocessing for reliable evaluation.*

### dataloader.py - DataLoader Creation

```
# Creates three dataloaders with proper splitting
train_loader, val_loader, test_loader = create_dataloaders(...)

# Split ratios: 70% train, 15% val, 15% test
```

```
# Stratified split ensures each set has proportional class representation
```

---

## 5.2 src/models/ - Neural Network

### classifier.py - Main Model

```
class LungCancerClassifier(nn.Module):
    """
    Architecture:
    +-------------------------------------------------------------+
    |                      ResNet-50 Backbone                     |
    |-------------------------------------------------------------|
    |  conv1 -> bn1 -> relu -> maxpool                              |
    |      v                                                      |
    |  layer1 (64 -> 256 channels)    - 3 Bottleneck blocks       |
    |      v                                                      |
    |  layer2 (256 -> 512 channels)   - 4 Bottleneck blocks       |
    |      v                                                      |
    |  layer3 (512 -> 1024 channels)  - 6 Bottleneck blocks       |
    |      v                                                      |
    |  layer4 (1024 -> 2048 channels) - 3 Bottleneck blocks <- Grad-CAM target
    |      v                                                      |
    |  avgpool (adaptive)                                         |
    |      v                                                      |
    |  Dropout(0.5)                                               |
    |      v                                                      |
    |  fc (2048 -> 5 classes)         - Modified for our task     |
    +-------------------------------------------------------------+
    """
```

Key Implementation Details:

```
# Load pretrained weights
weights = models.ResNet50_Weights.IMAGENET1K_V2
self.backbone = models.resnet50(weights=weights)

# Modify final layer for our task
# Original: 2048 -> 1000 (ImageNet)
# Modified: 2048 -> 5 (Our classes)
self.backbone.fc = nn.Sequential(
    nn.Dropout(p=0.5),
    nn.Linear(2048, 5)
)
```

### model_factory.py - Model Creation

```
def create_model(model_name, num_classes, pretrained, ...):
    """
    Factory pattern for creating models
    - Centralizes model creation
    - Easy to add new architectures
    - Consistent configuration
    """
```

---

## 5.3 src/xai/ - Explainable AI

### gradcam.py - Grad-CAM Implementation

What is Grad-CAM?

> *Gradient-weighted Class Activation Mapping - uses gradients flowing into the final convolutional layer to produce a heatmap highlighting important regions.*

How It Works (Step by Step):

```
Step 1: Forward Pass
        Input image -> Model -> Feature maps (A) captured at layer4

Step 2: Get Target Score
        Output logits -> Select score for target class (yc)

Step 3: Backward Pass
        Compute dyc/dA (gradients of target class w.r.t. feature maps)

Step 4: Global Average Pooling
        alphak = (1/Z) Sumi Sumj (dyc/dAkij)
        (Importance weight for each feature map k)

Step 5: Weighted Combination
        L = ReLU(Sumk alphak ? Ak)
        (Combine feature maps weighted by importance, keep only positive)

Step 6: Upscale
        Resize heatmap to input image size (224x224)
```

Implementation:

```
class GradCAM:
    def __init__(self, model, target_layer):
        # Register hooks to capture activations and gradients
        self.forward_handle = target_layer.register_forward_hook(forward_hook)
        self.backward_handle = target_layer.register_full_backward_hook(backward_hook)
```

```
    def generate(self, input_tensor, target_class=None):
        # 1. Forward pass
        output = self.model(input_tensor)

        # 2. Get target class score
        target_score = output[0, target_class]

        # 3. Backward pass
        target_score.backward()

        # 4. Compute weights (global average pooling of gradients)
        weights = self.gradients.mean(dim=[2, 3], keepdim=True)

        # 5. Weighted combination with ReLU
        cam = (weights * self.activations).sum(dim=1)
        cam = F.relu(cam)

        # 6. Normalize to [0, 1]
        cam = (cam - cam.min()) / (cam.max() - cam.min())

        return cam
```

Why layer4?

> *Layer4 is the last convolutional layer in ResNet-50. It has:*
>
> *- Highest semantic meaning (understands "what" is in the image)*
>
> *- 7x7 spatial resolution (enough for localization)*
>
> *- 2048 channels (rich feature representations)*

## visualize.py - Visualization Utilities

```
def create_heatmap_overlay(image, heatmap, alpha=0.4):
    """
    Overlays colored heatmap on original image

    Process:
    1. Normalize heatmap to [0, 255]
    2. Apply JET colormap (blue->green->yellow->red)
    3. Blend with original image (alpha blending)
    """

def visualize_gradcam(image, heatmap, predicted_class, confidence):
    """
    Creates 3-panel visualization:
    [Original Image] [Heatmap Only] [Overlay]
    """
```

# 5.4 src/rag/ - RAG Pipeline

## THIS IS THE NOVEL CONTRIBUTION

### knowledge_base.py - Medical Knowledge Store

```
class MedicalKnowledgeBase:
    """
    WHAT: Structured repository of medical knowledge

    WHY Local (not LLM)?
    |-- [OK] Controlled content - all facts are verified
    |-- [OK] No hallucination - deterministic responses
    |-- [OK] Citable - every fact has a source
    |-- [OK] Works offline - no API dependency
    +-- [OK] Explainable - can show exactly what knowledge is used

    STRUCTURE:
    {
        "id": "adeno_001",
        "keywords": ["adenocarcinoma", "peripheral", "outer"],
        "content": "Adenocarcinoma typically presents in peripheral...",
        "source": "Travis WD et al., WHO Classification, 2021"
    }
    """
```

Knowledge Retrieval Process:

```
Query: "adenocarcinoma peripheral upper"
    v
Tokenize query into keywords
    v
Search keyword index
    v
Rank by keyword matches
    v
Return top-k entries with sources
```

### xai_to_text.py - XAI-to-Text Conversion (NOVEL!)

```
class XAITextConverter:
    """
    THE BRIDGE FROM VISUAL TO TEXTUAL

    Input: Grad-CAM heatmap [H, W]
    Output: Textual description + Keywords for RAG
```

```
    Process:
    1. Analyze spatial distribution (divide into 3x3 grid)
    2. Find high-attention regions
    3. Determine location (peripheral/central, upper/lower)
    4. Quantify intensity (very_high, high, moderate, low)
    5. Generate natural language description
    6. Extract keywords for knowledge base query
    """
```

Spatial Analysis:

```
+-------------+-------------+-------------+
|  top_left   | top_center  |  top_right  |
|  (upper     |  (upper     |  (upper     |
|   left)     |   central)  |   right)    |
|-------------+-------------+-------------|
| middle_left |middle_center| middle_right|
|  (middle    |  (central)  |  (middle    |
|   left)     |             |   right)    |
|-------------+-------------+-------------|
| bottom_left |bottom_center| bottom_right|
|  (lower     |  (lower     |  (lower     |
|   left)     |   central)  |   right)    |
+-------------+-------------+-------------+

For each region: Calculate mean attention value
Determine: Which regions have highest attention?
Map to: Medical terminology (peripheral = outer regions, central = middle)
```

## explanation_generator.py - Final Explanation

```
class ExplanationGenerator:
    """
    Combines everything into coherent explanation

    Components:
    1. XAITextConverter - "Model focused on peripheral upper region"
    2. MedicalKnowledgeBase - "Adenocarcinoma typically presents peripherally"

    Output Format:
    +------------------------------------------------------------+
    | PREDICTION: Adenocarcinoma (92.3%)                         |
    |------------------------------------------------------------|
    | VISUAL EVIDENCE:                                           |
    | The model focused on the peripheral upper region with      |
    | high attention intensity, concentrated in specific areas.  |
    |------------------------------------------------------------|
    | MEDICAL CONTEXT:                                           |
    | Adenocarcinoma typically presents in the peripheral regions |
    | of the lung. Ground-glass opacity is frequently associated  |
```

```
    | with this cancer type.                                    |
    |-----------------------------------------------------------|
    | Sources: Travis WD et al., WHO Classification 2021        |
    +-----------------------------------------------------------+
    """
```

---

## 5.5 src/utils/ - Utilities

### config.py - Centralized Configuration

```
@dataclass
class Config:
    # Paths
    base_dir = "d:\\Major Project"
    dataset_dir = "archive (1)/Lung Cancer Dataset"

    # Classes
    class_names = ["adenocarcinoma", "Benign cases",
                   "large cell carcinoma", "Normal cases",
                   "squamous cell carcinoma"]
    num_classes = 5

    # Training
    image_size = (224, 224)
    batch_size = 32
    num_epochs = 10
    learning_rate = 1e-4

    # Model
    model_name = "resnet50"
    pretrained = True
    dropout_rate = 0.5
```

### metrics.py - Evaluation Metrics

```
# Metrics used:
accuracy_score()   # Overall correctness
precision_score()  # How many predicted positives are true
recall_score()     # How many actual positives were found
f1_score()         # Harmonic mean of precision & recall
confusion_matrix() # Full breakdown of predictions
```

Why Recall is Important in Medical Imaging:

> *Missing a cancer case (False Negative) is more dangerous than a false alarm (False Positive). We want*

*HIGH RECALL to catch all potential cancers.*

---

# 6. KEY ALGORITHMS & TECHNIQUES

## 6.1 Transfer Learning

Concept:

```
ImageNet (1.2M images, 1000 classes)
         v Pre-training
    ResNet-50 learns general features
    (edges, textures, shapes, objects)
         v Transfer
    Load these weights
         v Fine-tune
    Train on our lung cancer dataset
    (Only modify final layers significantly)
```

Why It Works:

> *Low-level features (edges, textures) learned on natural images transfer well to medical images. High-level features get adapted during fine-tuning.*

## 6.2 Residual Connections (ResNet)

Problem: Very deep networks suffer from vanishing gradients

Solution: Skip connections that add input to output

```
Traditional:      Residual:
x                 x ---------------------+
v                 v                      |
Conv              Conv                   |
v                 v                      |
Conv              Conv                   |
v                 v                      |
y                 y = F(x) + x <----------+
                      +-- Residual connection
```

Mathematical Form:

```
H(x) = F(x) + x

Where:
- x is the input
- F(x) is the learned residual
- H(x) is the output
```

## 6.3 Cross-Entropy Loss

Used for: Multi-class classification

Formula:

L = -\sum_{c=1}^{C} y_c \log(p_c)

Where:

- C = number of classes (5)
- y_c = 1 if correct class, 0 otherwise
- p_c = predicted probability for class c

## 6.4 AdamW Optimizer

Why AdamW?

- Adaptive learning rate per parameter
- Weight decay for regularization
- Works well with pretrained models

Key Parameters:

```
optimizer = torch.optim.AdamW(
    model.parameters(),
    lr=1e-4,           # Learning rate
    weight_decay=1e-4  # L2 regularization
)
```

---

# 7. DATASET DETAILS

## 7.1 Dataset Source

Kaggle Lung Cancer CT Scan Dataset

## 7.2 Dataset Structure

```
archive (1)/Lung Cancer Dataset/
|-- adenocarcinoma/          # ~1500 images
|-- Benign cases/            # ~1500 images
|-- large cell carcinoma/    # ~1500 images
|-- Normal cases/            # ~1500 images
+-- squamous cell carcinoma/ # ~1500 images
```

## 7.3 Image Characteristics

| Property | Value |
|---|---|
| Format | PNG/JPG |
| Original Size | Variable |
| Processed Size | 224x224 |
| Color | RGB (converted if grayscale) |
| Source | CT scan slices |

## 7.4 Data Split

| Split | Percentage | Purpose |
|---|---|---|
| Training | 70% | Learning model weights |
| Validation | 15% | Hyperparameter tuning, earl... |
| Test | 15% | Final unbiased evaluation |

Stratified Split: Ensures each split has proportional class representation

---

# 8. TRAINING PROCESS

## 8.1 Training Pipeline

```
for epoch in range(num_epochs):
    # Training Phase
    model.train()
    for batch in train_loader:
        1. Forward pass: predictions = model(images)
        2. Compute loss: loss = criterion(predictions, labels)
        3. Backward pass: loss.backward()
        4. Update weights: optimizer.step()

    # Validation Phase
    model.eval()
    with torch.no_grad():
        Evaluate on validation set
        Track validation accuracy

    # Save best model if validation improves
    if val_accuracy > best_accuracy:
        save_checkpoint(model, "best_model.pth")
```

## 8.2 Training Configuration

| Hyperparameter | Value | Reason |
|---|---|---|
| Epochs | 10 | Sufficient for convergence ... |
| Batch Size | 32 | Balance between memory and ... |
| Learning Rate | 1e-4 | Conservative for fine-tunin... |
| Optimizer | AdamW | Adaptive + weight decay |
| Loss | CrossEntropyLoss | Standard for multi-class |

## 8.3 Checkpoints Saved

| File | Description |
|---|---|
| best_model.pth | Model with highest validati... |
| final_model.pth | Model after all epochs comp... |

---

# 9. EVALUATION METRICS

## 9.1 Primary Metrics

### Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

### Precision (Per Class)

$$\text{Precision} = \frac{TP}{TP + FP}$$

> *"Of all predicted as X, how many are actually X?"*

### Recall (Per Class)

$$\text{Recall} = \frac{TP}{TP + FN}$$

> *"Of all actual X, how many did we find?"*

### F1-Score

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

> *Harmonic mean of precision and recall*

## 9.2 Confusion Matrix

```
                    Predicted
                  A    SCC   LCC   B    N
         A      [TP   ...   ...  ...  ...]
Actual   SCC    [...   TP   ...  ...  ...]
         LCC    [...  ...   TP   ...  ...]
         B      [...  ...  ...   TP  ...]
         N      [...  ...  ...  ...   TP]

Where: A=Adenocarcinoma, SCC=Squamous, LCC=Large Cell, B=Benign, N=Normal
```

---

# 10. POTENTIAL QUESTIONS & ANSWERS

## Architecture Questions

### Q1: Why ResNet-50 and not other architectures?

A: ResNet-50 offers the best balance for our use case:

- Proven medical imaging performance - widely used in literature
- Residual connections - solve vanishing gradient in deep networks
- 2048 features - rich representations for cancer patterns
- Grad-CAM compatible - layer4 produces clear, interpretable heatmaps
- Well-documented - extensive research papers and tutorials
- Transfer learning ready - ImageNet pretraining provides good initialization

### Q2: Why 224x224 image size?

A: This is the standard input size for ImageNet-pretrained models:

- The pretrained convolutional filters expect this resolution
- Changing size would require architecture modifications
- It provides sufficient detail for classification while being computationally efficient

### Q3: What are residual connections and why do they matter?

A: Residual connections add the input directly to the output: H(x) = F(x) + x

- Solves vanishing gradient: Gradients can flow directly through skip connections
- Enables depth: Without them, networks >20 layers become hard to train
- Identity mapping: If F(x)=0, the block simply passes input through

### Q4: Why transfer learning for medical imaging?

A: Medical datasets are typically small (thousands vs millions). Transfer learning helps because:

- Feature reuse: Low-level features (edges, textures) transfer across domains
- Better initialization: Pretrained weights start closer to optimal
- Faster convergence: Fewer epochs needed
- Reduced overfitting: More generalizable features

## Explainability Questions

### Q5: What is Grad-CAM and how does it work?

A: Grad-CAM (Gradient-weighted Class Activation Mapping) creates visual explanations:

- Forward pass: Get feature maps from target layer (layer4)
- Backward pass: Compute gradients of target class score w.r.t. feature maps
- Weight computation: Global average pool gradients to get importance weights
- Combine: Weighted sum of feature maps
- ReLU: Keep only positive activations (regions that help the prediction)
- Resize: Upscale to original image size

### Q6: Why Grad-CAM instead of other XAI methods?

A:

| Method | Pros | Cons | Why Not |
|---|---|---|---|
| Grad-CAM | No modification needed, cla... | Coarse resolution | [OK] Selected |
| LIME | Model-agnostic | Slow, many parameters | Too complex |
| SHAP | Theoretically sound | Computationally expensive | Too slow |
| Attention | Built-in for transformers | Requires architecture change | Not applicable to ResNet |

### Q7: What is the novel contribution - XAI-to-Text?

A: The XAI-to-Text converter bridges visual explanations with textual knowledge:

- Analyzes heatmap spatially - divides into 3x3 grid
- Identifies attention regions - peripheral, central, upper, lower
- Quantifies intensity - very high, high, moderate, low
- Generates keywords - "adenocarcinoma peripheral upper"
- Queries knowledge base - retrieves relevant medical facts

This is novel because it connects WHERE the model looks with WHY that matters medically.

## RAG Questions

### Q8: What is RAG and why use it?

A: RAG = Retrieval-Augmented Generation

- Concept: Instead of generating explanations from scratch, retrieve relevant facts from a curated knowledge base
- Why: Ensures accuracy, provides citations, no hallucination
- Implementation: Simple keyword matching (can upgrade to semantic search later)

### Q9: Why not use an LLM like GPT for explanations?

A:

| LLM Approach | Our Approach |
|---|---|
| Can hallucinate | Deterministic, verified facts |
| Black-box generation | Transparent retrieval |
| Needs API, costs money | Works offline, free |
| Hard to cite sources | Every fact has a citation |
| May be wrong | Content is curated by us |

For medical applications, reliability and citation are critical.

### Q10: How is the knowledge base structured?

A: JSON structure with:

```
{
  "id": "adeno_001",
  "keywords": ["adenocarcinoma", "peripheral", "outer"],
  "content": "Medical fact about adenocarcinoma...",
  "source": "Travis WD et al., WHO Classification, 2021"
}
```

- Keywords: Trigger terms for retrieval
- Content: Verified medical information
- Source: Academic citation

## Training Questions

### Q11: Why 70-15-15 split?

A: Standard split for ML projects:

- 70% Training: Main learning data
- 15% Validation: Tune hyperparameters, monitor for overfitting
- 15% Test: Final unbiased evaluation (never seen during training)

Stratified split ensures each set has proportional class distribution.

### Q12: What is data augmentation and why use it?

A: Data augmentation artificially increases training data variety:

- Horizontal flip: Lungs are roughly symmetric

- Rotation (+/-15 deg): Handles slight orientation variations
- Color jitter: Handles CT scanner variations

Why: Prevents overfitting, improves generalization, especially important with limited medical data.

### Q13: Why use dropout?

A: Dropout randomly sets neurons to 0 during training:

- Rate 0.5: 50% of neurons dropped each iteration
- Effect: Prevents co-adaptation of neurons
- Result: More robust features, better generalization

### Q14: Why AdamW optimizer?

A: AdamW = Adam with decoupled weight decay

- Adaptive learning rate: Different rate per parameter
- Momentum: Smooths gradient updates
- Weight decay: L2 regularization for better generalization
- Works well with pretrained models

## Metrics Questions

### Q15: Why is recall important in medical imaging?

A: In cancer detection:

- False Negative (missing cancer) = Dangerous, patient doesn't get treatment
- False Positive (false alarm) = Less dangerous, just additional tests

We want HIGH RECALL to minimize false negatives.

### Q16: What does the confusion matrix tell us?

A: Shows detailed breakdown:

- Diagonal = correct predictions
- Off-diagonal = where model confuses classes
- Helps identify: Which cancers are confused with each other?

## Technical Questions

## Q17: What are forward and backward hooks?

A: PyTorch hooks capture intermediate values:

- Forward hook: Captures output of a layer during forward pass
- Backward hook: Captures gradients during backward pass
- Usage in Grad-CAM: Capture feature maps and gradients for heatmap computation

## Q18: What is the model's output format?

A:

- Raw output: Logits (unnormalized scores) for each class
- After softmax: Probabilities summing to 1
- Prediction: argmax of probabilities

## Q19: How do you handle class imbalance?

A:

- Stratified split: Ensures proportional class distribution
- Weighted loss (optional): Give more weight to minority classes
- Data augmentation: Increases effective training data
- Metrics: Use macro-averaged F1 instead of accuracy

## Q20: What happens during inference (prediction)?

A:

```
1. Load image -> Preprocess (resize, normalize)
2. Forward pass -> Get logits
3. Softmax -> Get probabilities
4. Grad-CAM -> Get heatmap
5. XAI-to-Text -> Get description
6. RAG -> Get medical context
7. Combine -> Final explanation
```

---

# 11. QUICK REFERENCE CARDS

## 11.1 Pipeline Summary Card

```
+-----------------------------------------------------+
|              LUNGXAI PIPELINE SUMMARY                |
|-----------------------------------------------------|
| INPUT:  CT Scan Image (any size, PNG/JPG)           |
|         v                                           |
| PREPROCESS: Resize 224x224, Normalize               |
|         v                                           |
| MODEL: ResNet-50 (pretrained, fine-tuned)           |
|         v                                           |
| OUTPUT: 5-class probabilities                       |
|         v                                           |
| XAI: Grad-CAM heatmap (layer4)                      |
|         v                                           |
| CONVERT: XAI-to-Text (spatial analysis)             |
|         v                                           |
| RAG: Knowledge retrieval (keyword matching)         |
|         v                                           |
| FINAL: Visual + Textual explanation                 |
+-----------------------------------------------------+
```

## 11.2 Technology Quick Reference

| Component | Technology | File |
|---|---|---|
| Model | ResNet-50 + PyTorch | classifier.py |
| XAI | Grad-CAM | gradcam.py |
| Knowledge | JSON Knowledge Base | knowledge_base.py |
| Bridge | XAI-to-Text Converter | xai_to_text.py |
| Visualization | OpenCV + Matplotlib | visualize.py |

## 11.3 Key Numbers

| Metric | Value |
|---|---|
| Input Size | 224 x 224 x 3 |
| Model Parameters | ~25.6M (ResNet-50) |
| Feature Size | 2048 (before FC) |
| Output Classes | 5 |
| Grad-CAM Layer | layer4 (7x7x2048) |

| | |
|---|---|
| Dropout Rate | 0.5 |
| Learning Rate | 1e-4 |

## 11.4 Class Information Card

| Class | Typical CT Features | Typical Location |
|---|---|---|
| Adenocarcinoma | Ground-glass opacity, spicu... | Peripheral |
| Squamous Cell | Central mass, cavitation | Central/Hilar |
| Large Cell | Large peripheral mass | Peripheral |
| Benign | Well-defined margins, calci... | Variable |
| Normal | Clear lung fields | N/A |

## 11.5 Formula Quick Reference

```
Grad-CAM:    L = ReLU(Sumk alphak ? Ak)   where alphak = GAP(dyc/dAk)

Loss:        L = -Sum yc log(pc)

Accuracy:    TP + TN / Total

Precision:   TP / (TP + FP)

Recall:      TP / (TP + FN)

F1:          2 x (P x R) / (P + R)
```

---

# FINAL TIPS FOR REVIEW

- Start with the big picture - Explain the problem, then the solution
- Know the "why" - Be prepared to justify every technology choice
- Understand the flow - Be able to trace data through the entire pipeline
- Remember key numbers - 224x224, 5 classes, 2048 features, layer4
- Emphasize novelty - XAI-to-Text conversion is your unique contribution
- Clinical relevance - Always connect back to why this helps doctors

Good luck with your review!

---

Document generated for Major Project Review - January 2026