Babeş-Bolyai University
Faculty of Mathematics and Computer Science

Early Lung Cancer Detection Using Artificial Intelligence

Moghioroş Eric
Croitoru Andreea Bianca

Prof. Drd. Dioşan Laura
Drd. Limboi Sergiu

May 2025

Contents

1. Introduction

Lung cancer is one of the most lethal cancers globally, with a high mortality rate primarily caused by late detection. Early and accurate diagnosis plays a pivotal role in improving treatment outcomes and patient survival. Recent advances in artificial intelligence, particularly in deep learning, have paved the way for automated systems that can assist in medical image analysis with a performance nearing that of expert radiologists.

This project introduces a multi-stage deep learning pipeline designed to support the detection, classification, and localization of lung abnormalities in axial slices from chest CT scans. The pipeline employs 2D Convolutional Neural Networks (CNNs) in a modular architecture to replicate the diagnostic process used by clinicians.

The pipeline is composed of the following key components:

1. **Primary Classification Model:** A multi-class CNN that analyzes a given CT slice and classifies it into one of three categories: *Normal*, *Benign*, or *Malignant*. This model acts as the entry point of the pipeline and quickly filters out normal cases.

2. **Tumor Localization Model:** If the classification result is *Benign* or *Malignant*, the image is passed to a secondary CNN that performs object localization. This model predicts bounding box coordinates that delineate the tumor area within the image, thus providing spatial context and visual support for clinical interpretation.

By combining classification and localization, the system not only flags potentially pathological cases but also highlights the specific region of interest, making it a valuable decision-support tool for radiologists. Each model is trained and validated independently to optimize its performance and ensure robustness across diverse imaging conditions.

This modular design allows for flexible updates, such as replacing or fine-tuning individual components as more data becomes available or clinical needs evolve. Future sections will describe the dataset, preprocessing techniques, augmentation strategies, model architectures, training methodologies, evaluation metrics, and experimental results.

2. Dataset

2.1 Data Source

The dataset used in this project is the *IQ-OTH/NCCD - Lung Cancer Dataset* (The Iraq-Oncol ogy Teaching Hospital/National Center for Cancer Diseases), publicly available from The Cancer Imaging Archive (TCIA). [7]

This dataset was collected in the above-mentioned specialist hospitals over a period of three months in fall 2019. It includes CT scans (originally collected in DICOM format) of patients diagnosed with lung cancer in different stages, as well as healthy subjects.

Each scan contains several slices (from 80 to 200), each of them represents an image of the human chest with different sides and angles. The cases vary in gender, age, educational attainment, area of residence, and living status. Some of them are employees of the Iraqi ministries of Transport and Oil, others are farmers and gainers. Most of them come from places in the middle region of Iraq, particularly, the provinces of Baghdad, Wasit, Diyala, Salahuddin, and Babylon.

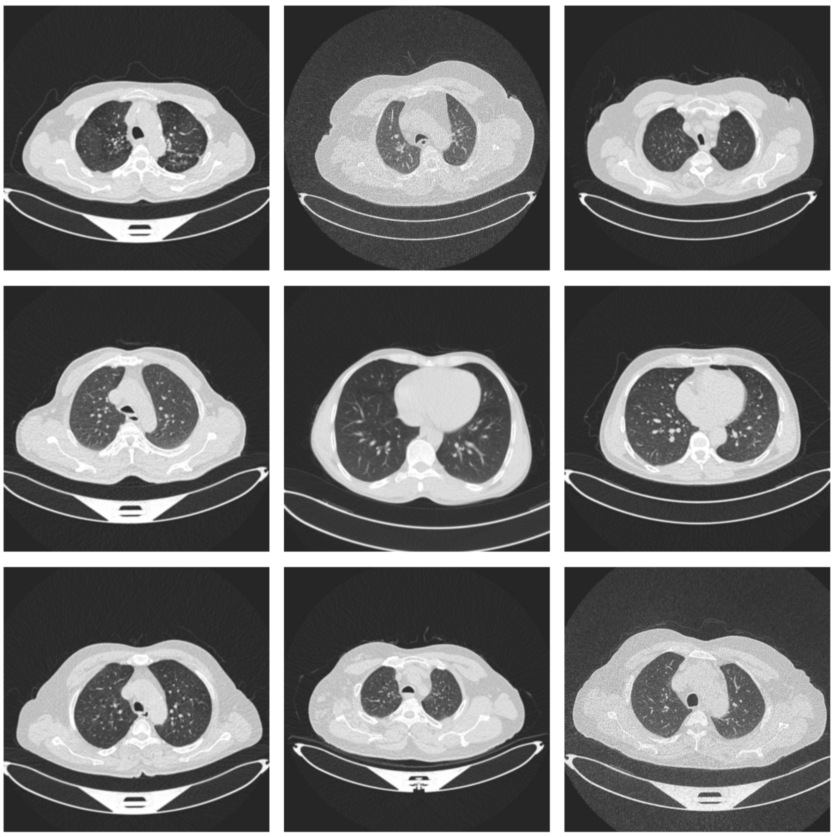


Figure 1. CT scan samples from dataset.

2.2 Data Description: Properties, Types and Formats

The IQ-OTH/NCCD lung cancer dataset consists of 3.609 chest CT scan images collected from 110 individual patients. Each image is stored in JPEG format with non-uniform resolutions.

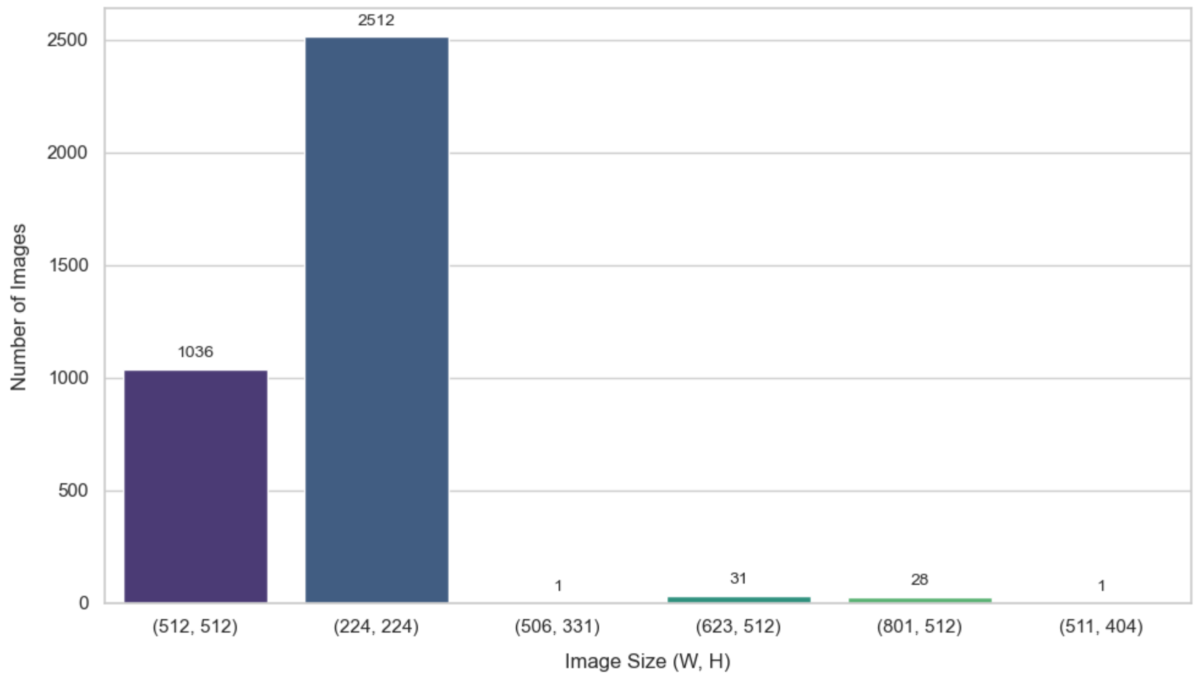


Figure 2. Distribution of images size.

2.3 Data Annotation: Methods and Categories

The dataset was manually organized into three distinct folders, each corresponding to a diagnostic category used for classification:

- **Normal** – images without visible nodules or abnormalities
- **Benign** – images containing nodules classified as non-cancerous
- **Malignant** – images containing nodules classified as cancerous

The labeling of each image was performed by a team of experienced oncologists and radiologists, ensuring a high degree of clinical accuracy and reliability in the annotation process. This expert-labeled dataset forms a strong foundation for training a supervised learning model in a sensitive medical context.

This folder-based organization enabled the model to correctly associate each image with its corresponding class label during training and evaluation. The distribution of these labeled samples is presented in Figure 3.

2.4 Data Preprocessing: Augumentation and Cleaning

As part of the preprocessing pipeline, non-square images were removed from the dataset prior to training. This decision was made to avoid potential issues associated with direct resizing or aspect-ratio-preserving padding, both of which can negatively impact the performance of CNN. To ensure consistency and compatibility with the input requirements of CNN, all remaining images were resized to a uniform resolution of 224×224 pixels.

Although techniques such as padding can preserve the original aspect ratio, they introduce artificial borders and non-informative regions into the image. These artifacts may be misinterpreted by the CNN as meaningful features, especially in the early convolutional layers, which are sensitive to spatial patterns. As a result, the model might learn irrelevant cues and generalize poorly, particularly in medical imaging tasks where precision is critical.

Furthermore, direct resizing of non-square images to a fixed square input size results in geometric distortion, altering the shape, size, and orientation of key anatomical structures such as lung nodules. This deformation can obscure clinically relevant patterns and lead to degraded model performance, especially in tasks involving subtle morphological differences between classes.

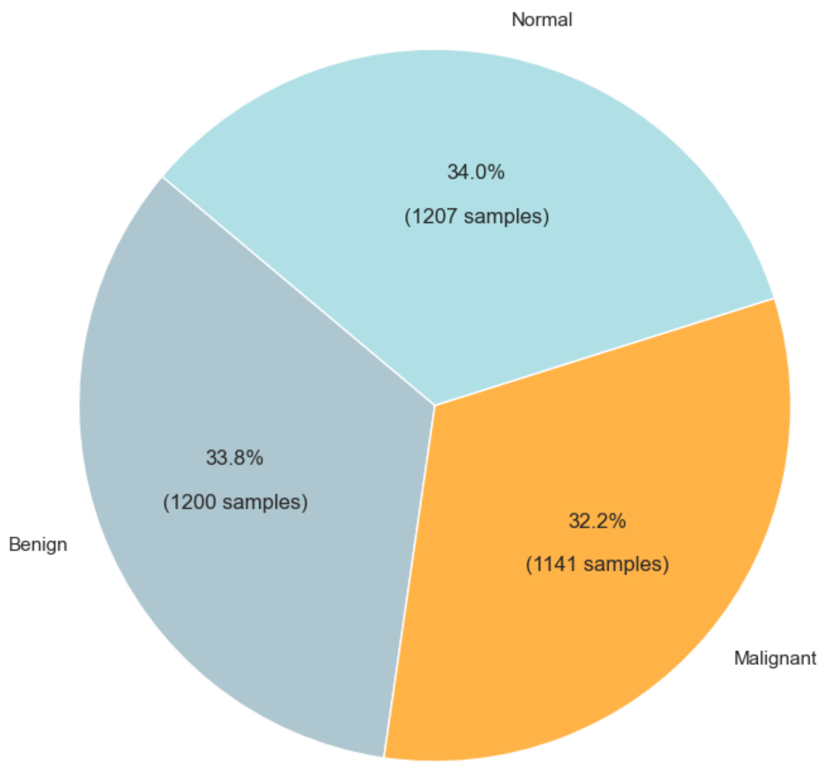


Figure 3. Distribution of images label after preprocessing.

The original dataset exhibited a significant imbalance among the diagnostic categories, which could adversely affect model training and lead to biased predictions. Specifically, the initial dataset included: 40 malignant cases, 15 benign cases and 55 normal (non-nodule) cases. To address this issue and ensure a more balanced distribution of classes, extensive data augmentation techniques to artificially expand the dataset were applied. The goal was to create a more uniform class distribution and enhance the model's ability to generalize across all categories.

The augmentation techniques applied include: Horizontal Flip, Vertical Flip, Rotation, Colorjitter, Contour Crop, Gaussian Blur, Sharpeness, Contrast and Histogram Equalization [7].

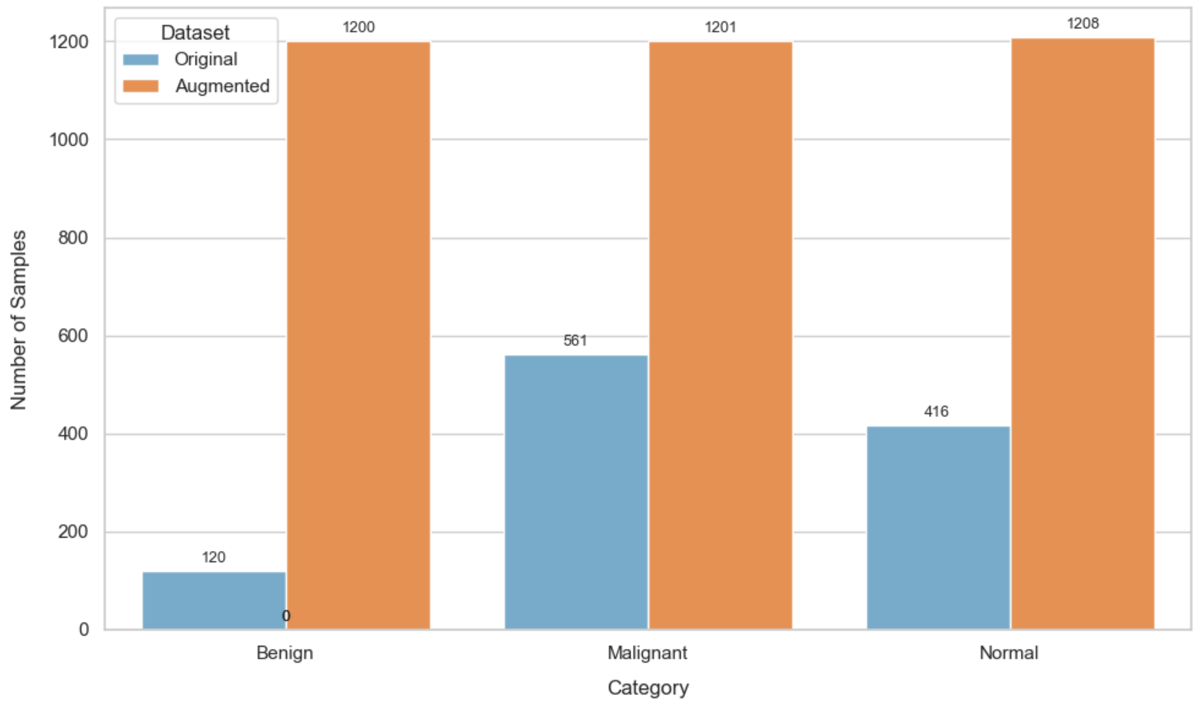


Figure 3. Distribution of images label compared to the original dataset.

2.5 Data Splitting: Training, Test and Validation

To ensure effective training, evaluation, and generalization of the Convolutional Neural Network (CNN), the dataset was split into three subsets using the following proportions: 75% Training Set, 15% Validation Set and 15% Test Set.

- **Effective Learning:** Allocating the majority of the data to the training set ensures that the model has sufficient samples to learn patterns, especially in a medical imaging context where inter-class variations may be subtle and complex.
- **Hyperparameter Tuning:** The validation set is used to monitor the model's

performance during training and to fine-tune hyperparameters. This helps prevent overfitting by evaluating how well the model generalizes to unseen data during the training process.

- **Unbiased Performance Evaluation:** The test set is strictly separated from the training and validation processes. It provides an objective estimate of the model's real-world performance, helping ensure that performance metrics are not inflated by exposure to the training data.

This division plays a crucial role in the training workflow and contributes to the development of a robust and unbiased model.

2.6 Challenges and Limitations

Despite its usefulness, the IQ-OTH/NCCD dataset presents several limitations:

- **Limited Dataset Size:** Only 3,609 CT images from 110 patients — a relatively small dataset — may lead to overfitting and reduced generalization.
- **Class Imbalance:** A severe initial imbalance, especially in benign cases, may affect model reliability, despite augmentation efforts.
- **Annotation Variability:** Expert annotations can still be subjective, with inter-observer variability potentially introducing label noise.
- **Lack of 3D Context:** Working with individual 2D slices disregards spatial continuity across scans, possibly omitting valuable diagnostic information.
- **Resolution Inconsistency:** JPEG compression artifacts and varying image quality hinder feature consistency and model accuracy.
- **Preprocessing Trade-offs:** Removal of non-square images and resizing may discard informative content or distort lesion morphology.
- **Geographic and Demographic Bias:** Patients from a specific region in Iraq limit generalizability across different populations and scanner environments.

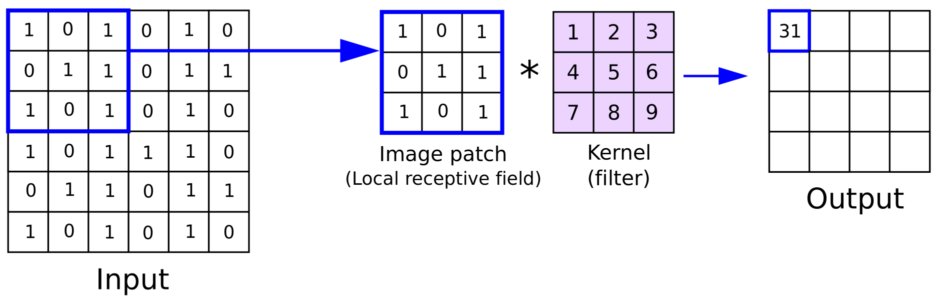
3. Model Architecture

Lung cancer detection using **Convolutional Neural Networks (CNNs)** has become a promising approach in medical imaging due to CNNs' ability to automatically learn and extract relevant features from complex CT scan images.

3.1 Overview of CNN structure (layers, activations)

- **Convolutional Layers:** These layers form the core of the CNN architecture. Each convolutional layer applies multiple filters (kernels) to the input or previous layer's feature maps to detect local patterns such as edges, textures, and shapes relevant to lung cancer.
  - **Filter size:** Typically 3×3 filters are used because they strike a balance between capturing fine-grained details and computational efficiency. [4]
  - **Number of filters:** The number of filters usually increases progressively through the network (e.g., 32, 64, 128, 256), allowing the model to learn from simple to complex features hierarchically.
  - **Stacked convolutions:** Multiple convolutional layers are stacked to deepen the network, enabling the detection of subtle malignancies and complex lung tissue patterns.

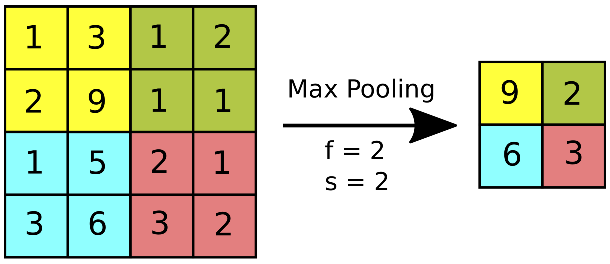
  The convolutional layers are vital for extracting meaningful features from the CT scans, which are critical for distinguishing between normal, benign, and malignant tissues.



Convolutional layer representation. [1]

- **Batch Normalization:** After each convolutional operation, batch normalization layers are inserted to normalize the activations. This technique stabilizes and accelerates training by reducing internal covariate shift, which helps the model converge faster and generalize better on unseen data. [4]

- **Activation Functions:** The Rectified Linear Unit (ReLU) activation function is applied after batch normalization in each convolutional block. ReLU introduces non-linearity by outputting zero for negative inputs and the input itself for positive values.
  - **Impact:** ReLU prevents the vanishing gradient problem, allowing deeper networks to learn effectively.
  - **Training efficiency:** It speeds up convergence during training.
  - **Feature learning:** Enables the network to learn complex, non-linear features essential for accurate lung cancer classification. [4]
- **Pooling Layers:** Max pooling layers with a typical pool size of 2×2 are inserted after convolutional blocks to downsample the feature maps.
  - **Purpose:** Reduce spatial dimensions, lowering computational load.
  - **Robustness:** Provides translation invariance, making the model less sensitive to small shifts or distortions in lung images.
  - **Information retention:** Max pooling preserves the most prominent features in each region, which is crucial for detecting cancerous nodules.



Max pooling representation. [1]

- **Dropout Layers:** Dropout layers randomly deactivate a fraction of neurons during training (commonly 0.5 dropout rate). This prevents the network from overfitting by forcing it to learn redundant and more robust feature representations. These layers are particularly important in medical imaging, where datasets may be limited and overfitting is a risk.
- **Flatten Layer:** After the convolutional and pooling layers, the multi-dimensional feature maps are flattened into a one-dimensional vector. This transformation prepares the data for the fully connected layers that perform the final classification.
- **Dense Layer:** One or more fully connected layers follow the flattening step. These layers integrate all extracted features to make a final decision on the presence and

type of lung cancer. Dense layers typically contain hundreds of neurons to capture complex feature interactions.

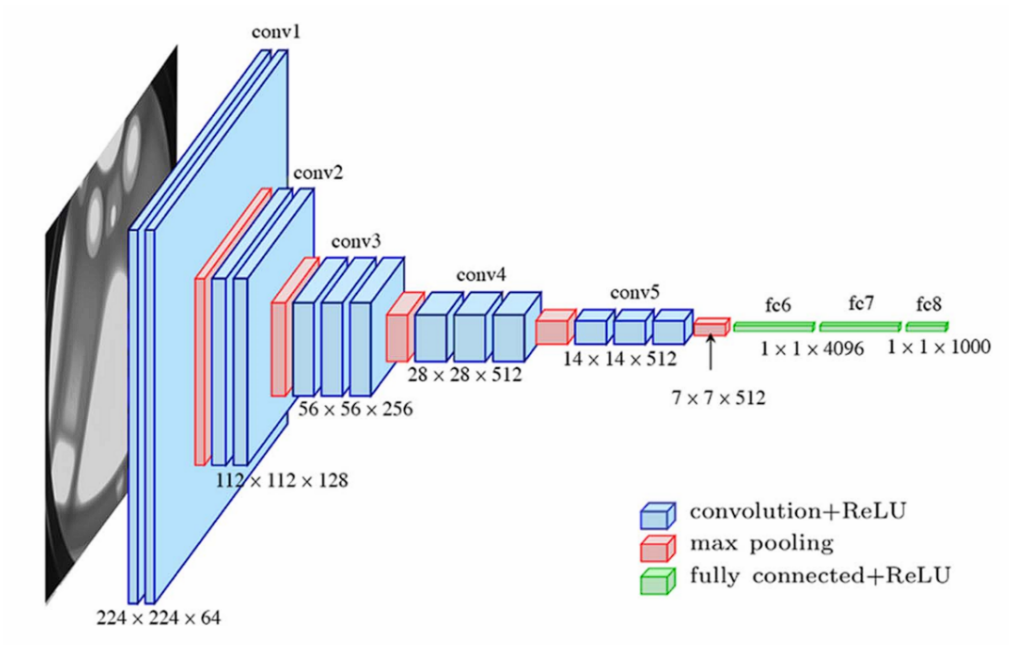


Diagram of Very Deep Convolutional Networks. [3]

3.2 Input/output format

- **Input Layer and Preprocessing:** The CNN model begins with an input layer designed to accept lung CT scan images resized to a standardized dimension, often 256×256 pixels. This resizing ensures consistent input size for the network and preserves critical spatial information necessary for detecting lung nodules or tumors. Preprocessing steps prior to input include noise reduction (techniques such as thresholding and segmentation are applied to remove irrelevant regions like bones, air, and background noise, isolating lung tissues for focused analysisand) and contrast enhancement to improve image quality while avoiding loss of important features. [5]
- **Output Layer and Prediction:**
  - **Binary classification:**
    - **Structure:** A single neuron with a sigmoid activation (e.g., cancerous vs. non-cancerous).
    - **Output:** A scalar value between 0 and 1 representing the probability that the input image contains cancerous tissue. A threshold (commonly

0.5) is applied to convert the probability into a class label (e.g., benign if probability less than 0.5, malignant otherwise).

- **Multiclass classification:**
  - **Structure:** Multiple neurons equal to the number of classes (e.g., normal, benign, malignant) with a softmax activation function.
  - **Output:** A probability distribution over all classes, where the sum of probabilities equals 1. The class with the highest probability is selected as the predicted label. For example, if the output is [0.1, 0.7, 0.2], the model predicts "benign" with 70% confidence.
- **Confidence Scores:** The output probabilities provide a measure of confidence in the prediction. This is essential for clinical applications where uncertain cases may require further examination or additional testing. Well-calibrated models produce probabilities that accurately reflect true likelihoods. Calibration techniques such as Platt scaling or isotonic regression may be applied post-training to improve reliability.

4. Implementation

ResNet, introduced by Kaiming He et al. in 2015, is a deep convolutional neural network architecture designed to overcome the vanishing gradient problem that hampers training of very deep networks. It achieves this by using residual learning through shortcut (skip) connections that allow gradients to flow directly across layers, enabling stable training of networks with dozens or even hundreds of layers without performance degradation. [2]

This architecture has been widely adopted in medical imaging tasks, including lung cancer detection, due to its ability to learn complex hierarchical features from images while maintaining training stability and accuracy. [6]

4.1 Environment and tools

The implementation of the lung cancer detection system was carried out using the *Python programming language*, chosen for its readability and wide support in the machine learning community. The model was developed and trained using the *PyTorch framework*, known for its dynamic computation graph and intuitive design. All experiments were conducted on a macOS environment using CPU-based processing, as no dedicated GPU was available. Despite the computational limitations, careful model optimization and efficient batch processing allowed for manageable training times. The overall environment provided a stable and flexible foundation for implementing and testing the ResNet50-based multiclass classification model.

4.2 Model setup

The model architecture for this project was based on **ResNet50**, a 50-layer deep convolutional neural network known for its use of residual connections to enable training of very deep networks. Using PyTorch's *torchvision.model*, a pre-trained version of ResNet50 was loaded, leveraging weights learned from the given dataset.

- **Convolutional Layers:** In ResNet, convolutional layers are organized within residual blocks and include a combination of 1×1 and 3×3 convolutions (bottleneck blocks) to efficiently capture complex features while controlling computational cost. They form the fundamental building blocks for feature extraction in the network.
- **Residual Blocks:** The hallmark of ResNet, residual blocks add the input of a block directly to its output via shortcut connections. This residual learning framework allows the network to learn the difference (residual) between the input and desired output, which helps prevent the vanishing gradient problem and enables training of very deep networks with improved accuracy and convergence.

- **Pooling Layer:** After the first 7×7 convolution layer and ReLU activation, a 3×3 max pooling with stride 2 is applied. Pooling reduces the number of parameters, adds translation invariance, and helps control overfitting.
- **Batch Normalization Layer:** Batch normalization is applied after each convolutional layer and before the activation (ReLU). It helps to standardize the inputs to each layer (zero mean and unit variance), making the model more stable.
- **Activation Functions:** The primary activation function used is ReLU (Rectified Linear Unit), defined as *f(x) = max(0, x).* ReLU is applied after each Batch Normalization layer in each convolutional block.
- **Fully Connected (Dense) Layer:** After all the convolutional and pooling layers, the output feature map is flattened into a vector. This vector is passed to a fully connected layer. The original ResNet50 has a Dense layer with 1000 outputs (for the 1000 ImageNet classes) followed by a softmax activation, but in our case we replaced the final dense layer with one that outputs 3 neurons, each representing a class (benign, malignant, normal).

5. Training

The training process was designed to fine-tune a pre-trained ResNet50 model for the task of multiclass classification (benign, malignant, normal). To ensure efficient and stable learning, several key hyperparameters were carefully selected.

5.1 Hyperparameters: epochs, learning rate, batch size

- **Epochs:** The training was conducted over **10 epochs**, meaning the entire dataset was passed through the model ten times. Ten epochs allow the model to learn the underlying patterns without excessive overfitting.
- **Learning rate:** This parameter, a critical one that controls how much the model weights are adjusted during training, was set to **0.0001**. This low learning rate was chosen to ensure gradual learning, particularly important when fine-tuning a pre-trained network, to avoid overwriting useful features learned from the source dataset.
- **Batch size:** A **batch size of 16** was used, which determines the number of samples processed before the model updates its weights. This relatively small batch size was chosen to accommodate CPU-based training and to balance between convergence speed and memory efficiency.

5.2 Loss function and optimizer

- **Loss function:** The loss function used was **cross-entropy loss**, which is standard for multi-class classification problems. It measures the performance of the classification model by comparing the predicted probabilities to the true class labels, and it encourages the model to assign high probability to the correct class.

$$\text{Loss}_{\text{CE}} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

- **Optimizer:** For optimization, the **Adam optimizer** was used, which adapts the learning rate for each parameter, combining the advantages of both AdaGrad and RMSProp. This optimizer is particularly well-suited for complex models and noisy gradients, often found in medical image data.

6. Results

6.1 Training Time and Performance

The lung cancer detection model, based on the ResNet50 architecture, was trained on a labeled dataset of medical lung images. The training process was conducted using two different hardware configurations: a GPU-enabled environment and a traditional CPU-based environment.

Hardware	Total Training Time	Average Time/Epoch
CPU (Apple M2, 10 cores)	5 hours 20 minutes	12.8 minutes
GPU (MPS)	48 minutes	1.92 minutes

Table 1: Training Time Comparison over 20 Epochs

While the CPU successfully completed the training task, it required several hours due to the computational intensity of deep convolutional neural networks like ResNet50. This highlights the limitations of using CPUs for large-scale deep learning tasks, particularly in terms of training time. The GPU significantly outperforms the CPU in terms of training time. This speedup allows for faster experimentation and model iteration.

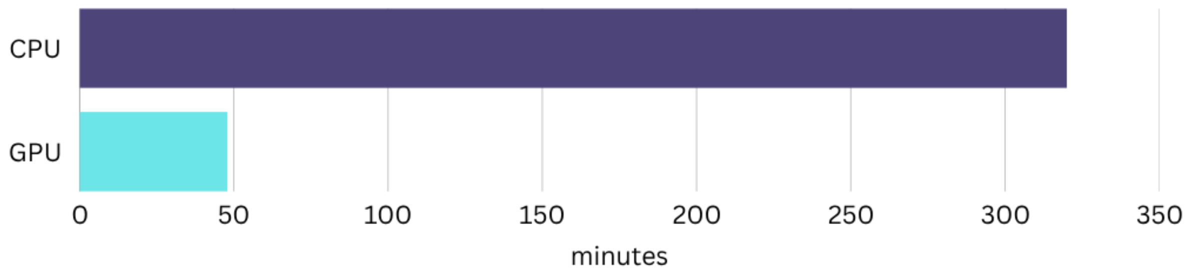


Chart comparison of training time between GPU and CPU.

6.2 Energy and CO_2 Footprint Analysis

To further understand the environmental and economic implications, we compared the energy consumption and estimated CO_2 emissions of the two setups.

Hardware	Power Usage	Time	Energy	CO_2 Emission (kg)
CPU	65 W	5.33 h	0.346	≈0.277
GPU	320 W	0.8 h	0.256	≈0.205

Table 2: Energy Consumption and CO_2 Emission Estimates

Assumptions:

- CO_2 emission factor: 0.8 kg CO_2 per kWh (based on global average).
- Electricity cost: \$0.15 per kWh (average in the US).

6.3 Cost Calculation

- **CPU Total Cost** = 0.346 kWh × \$0.15 = **\$0.0519**
- **GPU Total Cost** = 0.256 kWh × \$0.15 = **\$0.0384**

6.4 Theoretical GPU Performance (Compared to CPU)

Although the training was conducted on a CPU, it is important to consider the theoretical performance advantages of GPUs. Graphics Processing Units (GPUs) are highly optimized for parallel floating-point computations, which are common in deep learning tasks. The performance of GPUs is often measured in TFLOPS (Tera Floating Point Operations Per Second).

Effective TFLOPS Calculation

$$\text{FLOPS} = \text{Number of Cores} \times \text{Clock Speed (Hz)} \times \text{FLOPs per Cycle}$$

$$\text{TFLOPS} = \frac{\text{FLOPS}}{10^{12}}$$

For example, a GPU with:

- 8704 MPS cores
- Clock speed of 1.7 GHz
- 2 FLOPs per core per cycle

would yield:

$$\text{FLOPS} = 8704 \times 1.7 \times 10^9 \times 2 = 29.6 \times 10^{12} = 29.6 \text{ TFLOPS}$$

This is several orders of magnitude higher than a standard CPU, which may reach only up to 0.5–1.5 TFLOPS under optimal conditions.

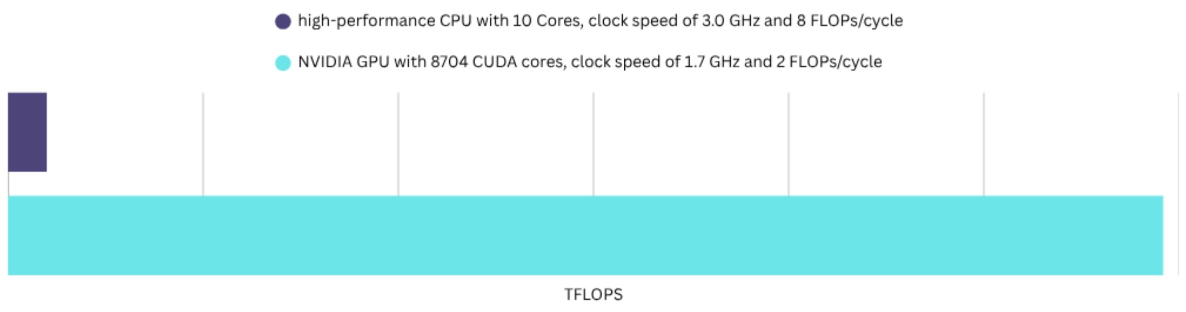


Chart comparison of TFLOPS values between GPU and CPU.

6.5 Components Influencing Training Speed

Several hardware and software factors significantly influence deep learning training performance:

- **Core Count and Parallelism** – More cores allow parallel execution of matrix operations.
- **Clock Speed** – Higher frequencies can increase operation speed, but also power consumption.
- **Memory Bandwidth** – High-speed memory access is critical for transferring large tensors during training.
- **Cache Size and Architecture** – Affects the ability to reuse data without frequent memory access.
- **Software Optimization** – Efficient usage of libraries such as PyTorch with multithreading and vectorization can impact performance greatly.

6.6 Summary of GPU vs CPU Comparison

- **Training Speed:** GPU offers a 6.7x speed improvement over CPU.
- **Energy Efficiency:** Despite higher power usage, GPU finishes faster, leading to lower overall energy consumption.
- **Cost-Effective:** GPU-based training costs less in terms of energy bills.
- **CO_2 Emissions:** GPU results in a slightly lower carbon footprint.

References

[1] Anh. Reynolds. *Convolutional Neural Networks (CNNs)*. 2019. URL: `https://anhreynolds.com/blogs/cnn.html`.

[2] Gaudenz Boesch. *Deep Residual Networks (ResNet, ResNet-50) A Complete Guide*. 2023. URL: `https://viso.ai/deep-learning/resnet-residual-neural-network/`.

[3] Gaudenz Boesch. *Very Deep Convolutional Networks (VGG) Essential Guide*. 2021. URL: `https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/`.

[4] Hammad, M., ElAffendi, M., El-Latif, A.A.A. et al. “Explainable AI for lung cancer detection via a custom CNN on CT images”. In: *Sci Rep* 15.12707 (2025).

[5] Pathan, Sameena et al. “An optimized convolutional neural network architecture for lung cancer detection.” In: *APL bioengineering* 8.2 (2024). DOI: `10.1063/5.0208520`.

[6] Seung Hyun Kim, Ho Chul Kang. “A Study on the Classification of Cancers with Lung Cancer Pathological Images Using Deep Neural Networks and Self-Attention Structures”. In: *Journal of Population Therapeutics and Clinical Pharmacology* 30.6 (2023).

[7] Subhajeet Das. *IQ-OTH/NCCD Lung Cancer Dataset (Augmented)*. 2025. DOI: `10.34740/KAGGLE/DS/6582139`. URL: `https://www.kaggle.com/ds/6582139`.