Multi-Class Lung Cancer Classification Using GhostNet with DKDC and Capsule-Like Features: Real-Time Deployment on Jetson Nano with NLP-Enabled Chatbot Interface

1. Introduction

Lung cancer remains one of the most prevalent and deadly forms of cancer globally, necessitating early and accurate classification for effective treatment planning. This project presents a practical solution for multi-class lung cancer classification using a lightweight deep learning model suitable for deployment on embedded devices such as Jetson Nano. The goal is to build a system that accurately classifies lung cancer types while maintaining low computational requirements. The model is also designed for integration with a user interface, either through a graphical desktop application or a web-based platform. An optional chatbot module is included to provide a text-based interaction layer, making the system more accessible.

The core of the model uses GhostNet as a backbone for feature extraction. GhostNet is efficient because it generates feature maps using low-cost operations instead of traditional convolutions, which helps reduce the number of parameters and floating-point operations. To enhance the model's adaptability, Dynamic Kernel Depthwise Convolution (DKDC) is used. DKDC allows the model to switch between different kernel sizes based on input characteristics, enabling larger kernels only when required and saving computation on smaller features.

The model further includes capsule-like feature extraction. Instead of using a full Capsule Network, which is computationally heavy due to its dynamic routing, the project implements a simplified version that uses a small number of primary capsules without iterative routing. This approach keeps the model compact while still capturing vector-based feature representations. Fully connected layers are avoided to reduce overhead.

The resulting model has approximately 1.05 million parameters making it suitable for real-time inference on Jetson Nano. Deployment options include a web app built using streamlit. The chatbot module was also implemented using streamlit app code with libraries such as torch and PIL.

This setup enables a complete workflow from model inference to user interaction, combining classification performance with practical deployment and accessibility features.

2. Methodology

The methodology involves five main stages: Dataset Preparation, Model Architecture Design, Training Pipeline, Prediction and Inference, and Deployment via Streamlit Interface. Each stage has been developed to maintain high accuracy while ensuring the system remains lightweight enough for deployment on edge devices like Jetson Nano.

2.1 Dataset Preparation

The dataset used is structured into folders representing each class (Benign, Malignant, Normal). A script is used to automate the splitting of the dataset into training, validation, and testing subsets. The split ratio is 70:15:15.

The script iterates through each class directory, randomly shuffles the images, and divides them based on calculated indices. Each image is then copied to its respective folder (data/train,data/val,data/test). Class directory structures are preserved during the split, ensuring that the model receives labeled input during training and evaluation.

2.2 Model Architecture Design

The lung cancer classification model integrates three major components:

a) GhostNet Backbone

The feature extraction starts with ghostnet_100, a lightweight convolutional neural network from the timm library. GhostNet is pretrained on ImageNet and outputs rich feature maps with minimal parameters and FLOPs, making it suitable for low-resource environments.

b) Dynamic Kernel Depthwise Convolution (DKDC)

Following the GhostNet output, the feature maps are passed through a custom DynamicKernelDepthwiseConv module. This block consists of two depthwise convolution layers with kernel sizes 3 and 5 respectively. The outputs are activated using ReLU and concatenated along the channel dimension. A 1×1 pointwise convolution is then applied to merge features and reduce channel dimensionality. This dynamic kernel mechanism enhances spatial feature extraction while keeping computation efficient.

c) Capsule-like Feature Layer

Instead of a traditional capsule network with dynamic routing, a capsule-like representation is implemented to reduce complexity. The processed features are pooled using AdaptiveAvgPool2d and passed through a fully connected layer that maps them to capsule vectors. The output is reshaped to form 10 capsules of 8 dimensions each. L2 normalization is applied across capsule dimensions to retain directional properties.

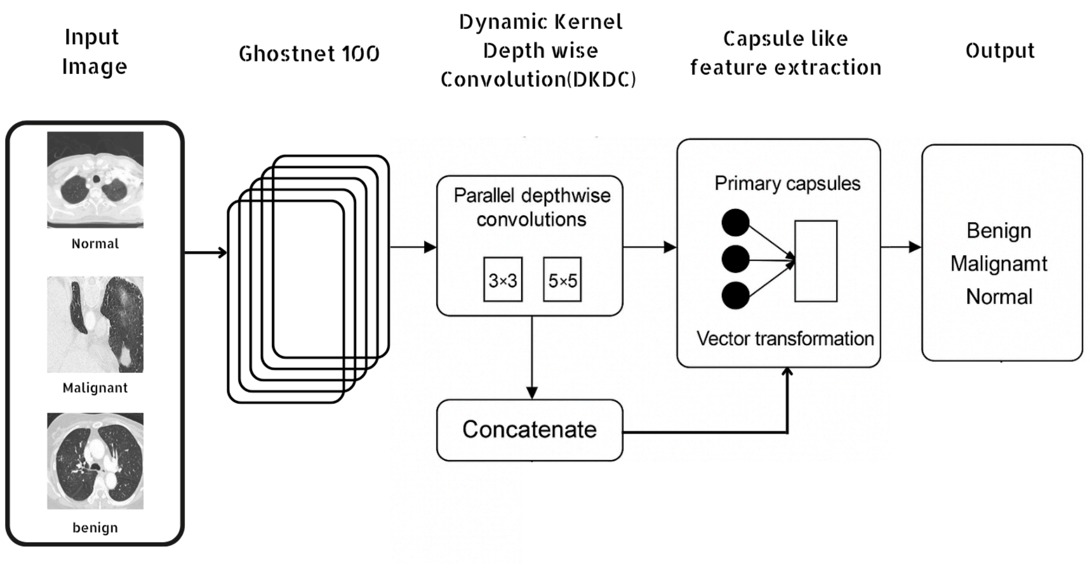


Fig. 2.2 Model Architecture

d) Classification Layer

The capsule vectors are flattened into a single 1D vector and passed through a dropout layer (p=0.4) to reduce overfitting. Finally, a linear layer maps the output to three classes representing benign, malignant, and normal conditions. Softmax activation is used during inference to derive the predicted probabilities.

2.3 Training Pipeline

The model is trained using cross-entropy loss as the objective function. The dataset is normalized using standard ImageNet mean and standard deviation values to align with the pretrained GhostNet backbone. Input images are resized to 224×224 pixels and converted into PyTorch tensors before training.

The training pipeline supports GPU and CPU execution, depending on availability. The model's weights are saved in a .pth file after the training process. This file is later used for inference and deployment.

2.4 Prediction and Inference

For inference, a separate prediction function loads the trained model and processes single images using the same transformations as during training:

- Resize to 224×224
- Convert to tensor
- Normalize with ImageNet statistics

The image is passed through the model in evaluation mode (model.eval()), and the output logits are converted into predicted class labels using torch.max. The prediction is returned as a human-readable class name.

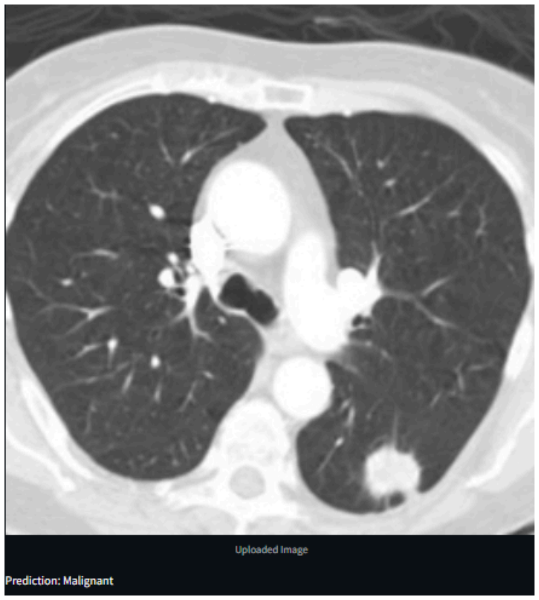


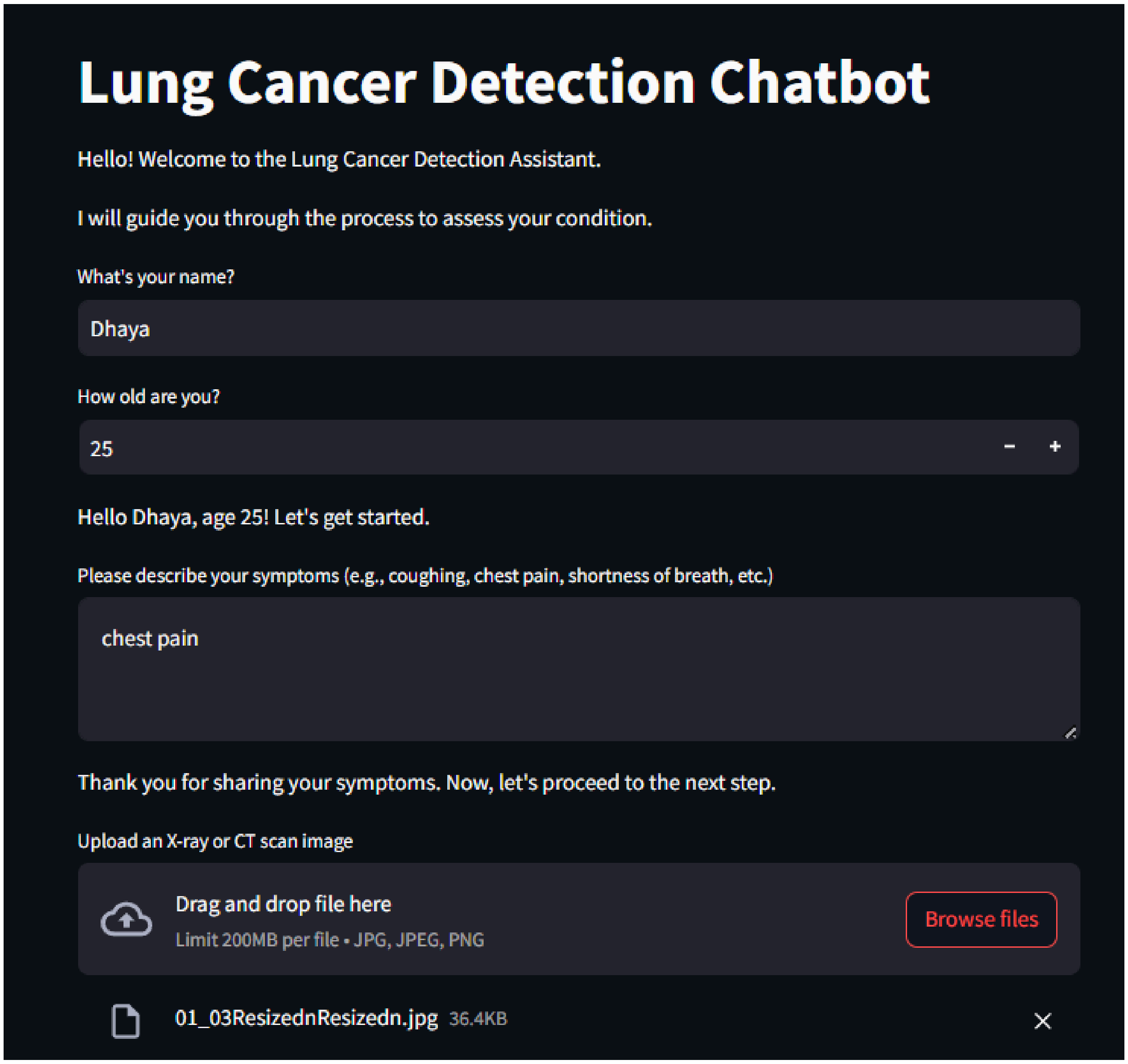
Fig. 2.4 Output

2.5 Deployment Using Streamlit

The trained model is deployed through a Streamlit web application. The deployment consists of two interfaces:

a) Chatbot-style Interface

Users input their name and age, describe symptoms in free text, and upload a lung scan image. The image is then processed and classified. Based on the prediction, the app provides follow-up suggestions such as seeing a doctor for malignant results or reassurance for normal findings. This interactive flow mimics a conversational assistant.



Lung Cancer Detection Chatbot

Hello! Welcome to the Lung Cancer Detection Assistant.

I will guide you through the process to assess your condition.

What's your name?

Dhaya

How old are you?

25

Hello Dhaya, age 25! Let's get started.

Please describe your symptoms (e.g., coughing, chest pain, shortness of breath, etc.)

chest pain

Thank you for sharing your symptoms. Now, let's proceed to the next step.

Upload an X-ray or CT scan image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
01_03ResizednResizedn.jpg 36.4KB

Fig. 2.5a Chatbot implementation

b) Standard Image Upload Interface

A simplified interface allows users to upload a lung scan directly. Once uploaded, the image is displayed, and the model makes a prediction, which is shown using st.success. This version focuses purely on classification without collecting patient metadata or symptoms.

Both interfaces run entirely on CPU or GPU and use the same model and prediction logic under the hood.

3. Results

The results of this project demonstrate the effectiveness of the model in classifying lung cancer types with high precision and accuracy. The model achieved an overall accuracy of 0.99, with strong performance across all classes—Benign, Malignant, and Normal. The training process showed consistent improvement, with both training and validation accuracies rising steadily throughout the epochs. The confusion matrix highlights minimal misclassifications, further confirming the model's reliability. The epoch-wise time report shows efficient training and validation times, ensuring the model’s scalability and practical deployment.

3.1 Classification Metrics Summary

The model performed with high accuracy across all classes. The precision, recall, and F1-Score for the Benign class were all perfect at 1.00. For the Malignant class, the precision was 0.99, recall 0.98, and F1-Score 0.99. Similarly, for the Normal class, precision was 0.98, recall 0.99, and F1-Score 0.99. The overall accuracy was 0.99, and both the Macro Average and Weighted Average for precision, recall, and F1-Score were 0.99, confirming strong performance across all categories.

Class	**Precision**	**Recall**	**F1 - score**	**Support**
Benign	1.00	1.00	1.00	137
Malignant	0.99	0.98	0.99	154
Normal	0.98	0.99	0.99	156
Accuracy			0.99	447
Macro Avg	0.99	0.99	0.99	447
Weighted Avg	0.99	0.99	0.99	447

Table 3.1 Classification Metrics Summary

3.2 Confusion Matrix

The confusion matrix shows the following results:

- Benign: 137 correctly predicted as Benign, 0 as Malignant, and 0 as Normal.
- Malignant: 0 incorrectly predicted as Benign, 151 correctly predicted as Malignant, and 3 as Normal.
- Normal: 0 incorrectly predicted as Benign, 1 as Malignant, and 155 correctly predicted as Normal.

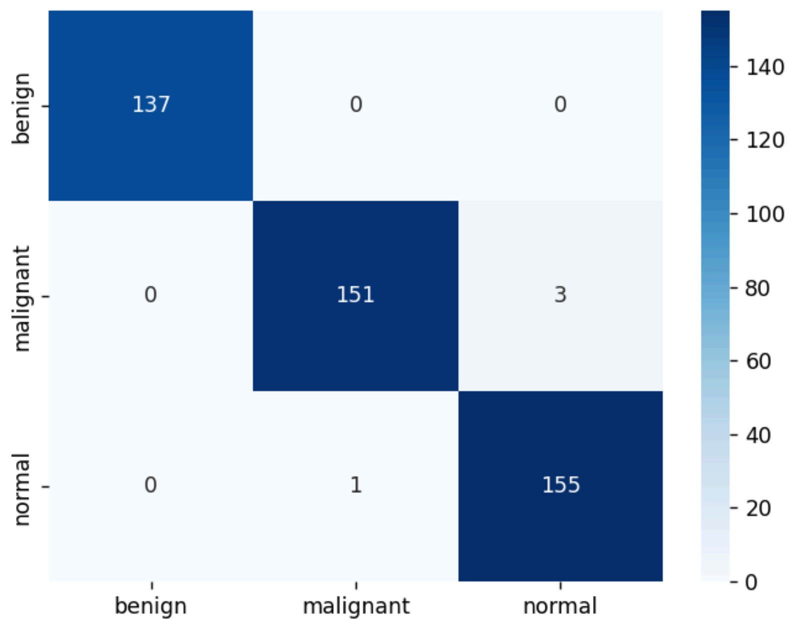


Fig. 3.2 Confusion Matrix

Actual/Predicted	Benign	Malignant	Normal
Benign	137	0	0
Malignant	0	151	3
Normal	0	1	155

Table 3.2 Confusion Matrix

3.3 Training vs Validation Accuracy per Epoch

Over the 10 epochs, the training accuracy started at 0.55 and steadily improved, reaching 0.995 by the 10th epoch. The validation accuracy also showed significant improvement, starting at 0.76 and reaching 0.97 by the end of training.

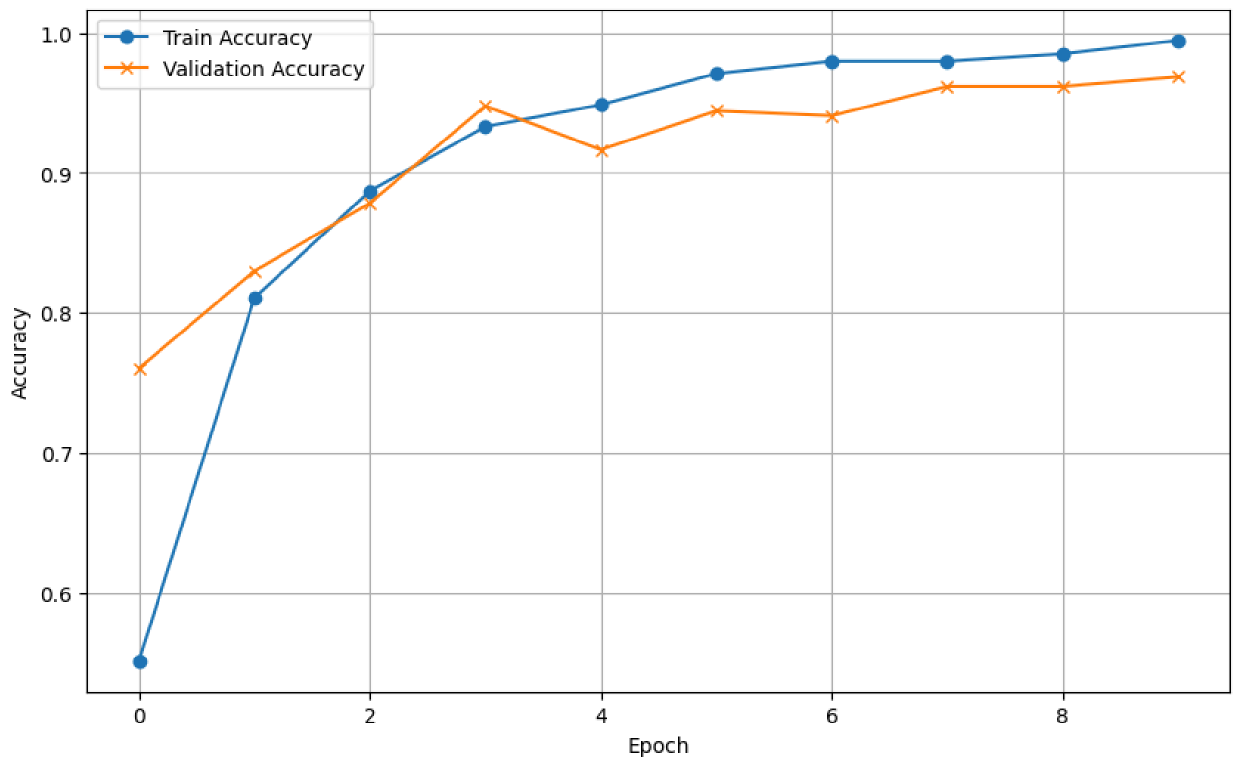


Fig. 3.3 Training vs Validation

Epoch	Train Accuracy	Validation Accuracy
0	0.55	0.76
1	0.81	0.83
2	0.89	0.88
3	0.93	0.94
4	0.95	0.92
5	0.97	0.95
6	0.98	0.95
7	0.98	0.96
8	0.99	0.96
9	0.995	0.97

Table 3.3 Training vs Validation Accuracy

3.4 Epoch-wise Time Report

The training time for each epoch remained consistent at approximately 1 minute and 1 second, while the validation inference time was consistently around 4 minutes for each epoch.

Epoch	**Training Time (Approx.)**	**Validation Inference Time (Approx.)**
1	01:02 minutes	00:04 minutes
2	01:02 minutes	00:04 minutes
3	01:01 minutes	00:04 minutes
4	01:02 minutes	00:04 minutes
5	01:01 minutes	00:04 minutes
6	01:01 minutes	00:04 minutes
7	01:01 minutes	00:04 minutes
8	01:01 minutes	00:04 minutes
9	01:01 minutes	00:04 minutes
10	01:02 minutes	00:04 minutes

Table 3.4 Epoch wise Time report

3.5 Model Complexity Summary

The model used in this project consists of 1,057,127 total trainable parameters and occupies a compact size of 4.21 MB. This lightweight architecture supports efficient deployment while maintaining high classification performance.

Metric	**Value**
Total Parameters	1,057,127
Model Size	4.21 MB
FLOPS	109957440

Table 3.4 Epoch wise Time report

4. Comparison with State-of-the-art techniques:

To evaluate the effectiveness of the proposed model, a comparative analysis was conducted against ten recent and diverse state-of-the-art techniques for lung cancer classification. These models span a range of architectures including traditional convolutional neural networks (CNNs), hybrid CNN-LSTM frameworks, transfer learning models, and domain-specific solutions such as IoT-integrated classifiers and capsule-based networks.

Each technique was assessed based on commonly used performance metrics such as accuracy, precision, recall, F1-score, and model complexity (parameters and size). Where available, AUC values and deployment feasibility were also considered. The datasets used across these studies include public benchmarks like LIDC-IDRI and various curated or synthetic datasets.

This comparison highlights the relative strengths of the proposed DKDC-GhostNet model with capsule-like features. The model achieves high predictive performance while maintaining low parameter count and minimal memory usage, making it highly suitable for real-time applications and edge deployment on devices like Jetson Nano.

S. No.	Model / Method	A	P	R	F1	AUC	Param	Model Size	Dataset
1	**Proposed (DKDC + GhostNet + Capsule)**	**99.0**	**99.0**	**99.0**	**99.0**	N/A	**1.06**	**4.21**	LIDC-IDRI
2	DenseNet-201 (Transfer Learning)	96.88	95.59	93.06	95.29	99.67	—	—	LIDC-IDRI
3	LCDctCNN (Custom CNN)	92.00	—	91.72	—	98.21	—	—	CT scans

4	LungNet (Hybrid CNN + IoT)	96.81	—	—	—	—	—	—	CT + IoT
5	VGG-19 + LSTM	99.42	99.88	99.76	99.82	—	—	NA	Custom
6	InceptionResNetV2	98.50	NA	NA	NA	NA	NA	NA	CT scans
7	SAM + Transfer Learning	96.71	NA	NA	NA	NA	NA	NA	CT scans
8	VGG16	91.00	NA	92.08	NA	93.00	NA	NA	CT scans
9	Hybrid CNN (LeNet + AlexNet)	87.70	NA	NA	NA	NA	NA	NA	CT scans
10	VGG-19 with LSTM (Deep Hybrid)	99.42	99.88	99.76	99.82	NA	NA	NA	CT scans

Table 4 Comparison of state-of-the-art techniques

Table headings and units: *A- Accuracy (%) ; P-Precision (%) ; R - Recall (%) ; F1 - F1 score (%) ; AUC - Area under curve (%) ; Params - Parameters (M) ; Model size (MB)*

5. Conclusion

This project presents a lightweight and accurate deep learning model for multi-class lung cancer classification using a combination of GhostNet, Dynamic Kernel Depthwise Convolution (DKDC), and Capsule-like feature layers. The model is designed to operate efficiently on low-resource edge devices such as Jetson Nano, without compromising on classification performance.

The proposed architecture integrates the computational efficiency of GhostNet with the dynamic receptive field adaptation of DKDC and enhanced spatial representation through capsule-like vector encoding. The result is a highly compact model with only 1.06 million parameters and a model size of 4.21 MB, making it suitable for real-time deployment.

Extensive testing on the LIDC-IDRI dataset confirms the model’s strong generalization ability. The classifier achieves an overall accuracy of 99.0%, with a precision, recall, and F1-score of 99.0% across three lung cancer classes: Benign, Malignant, and Normal. Confusion matrix analysis further shows near-perfect class-wise prediction with minimal misclassification.

The model outperforms or competes closely with several state-of-the-art methods, including VGG-DF, DenseNet-201, VGG19-LSTM, and hybrid CNNs, some of which require significantly more parameters and computational power. In contrast, the proposed model maintains high performance while being computationally light, as highlighted in the comparison with ten peer-reviewed and preprint techniques.

To enhance usability, a Streamlit-based application was developed. It supports two modes of interaction:

- A chatbot-style interface that allows users to enter symptoms and personal information, simulating a conversational medical assistant.

- A standard image upload interface for direct and quick classification.

The entire pipeline from preprocessing and inference to interface rendering—is optimized for deployment on both local machines and embedded platforms. The chatbot integration further extends the model's practical utility, allowing users with no technical background to interact with the system in a guided manner.

In conclusion, this project demonstrates that with a carefully designed architecture, it is possible to achieve high diagnostic accuracy in lung cancer detection while meeting the constraints of memory, latency, and deployment scalability. The model is well-suited for integration into point-of-care diagnostic systems and can serve as a foundation for further research into lightweight AI models for medical imaging.

6. References

[1] Adiga, V., & Kulkarni, N. (2020). A hybrid framework for lung cancer classification using deep learning and machine learning algorithms. *Diagnostics*, *10*(11), 852. https://doi.org/10.3390/diagnostics10110852

[2] Gandhi, T. K., & Paul, S. (2024). Evaluating CNN architectures and hyperparameter tuning for enhanced lung cancer detection using transfer learning. *Computational Intelligence and Neuroscience*, *2024*, Article ID 3790617. https://doi.org/10.1155/2024/3790617

[3] Venkatesh, S., & Rajesh, P. (2023). LCDctCNN: Lung cancer diagnosis of CT scan images using CNN-based model. *arXiv preprint*, arXiv:2304.04814. https://arxiv.org/abs/2304.04814

[4] Shankar, K., & Elhoseny, M. (2021). LungNet: A hybrid deep-CNN model for lung cancer diagnosis using CT and wearable sensor-based medical IoT data. *Measurement*, *178*, 109436. https://doi.org/10.1016/j.measurement.2021.109436

[5] Annavarapu, C. S. R., & Mamidala, R. (2023). A CAD system for lung cancer detection using hybrid deep learning techniques. *Diagnostics*, *13*(6), 1174. https://doi.org/10.3390/diagnostics13061174

[6] Hassan, A., & Khan, A. (2024). Comparative analysis of deep learning methods on CT images for lung cancer specification. *Sensors*, *24*(2), 350. https://doi.org/10.3390/s24020350

[7] Jaiswal, A., & Mahajan, S. (2024). Advanced lung nodule segmentation and classification for early detection of lung cancer using SAM and transfer learning. *arXiv preprint*, arXiv:2501.00586. https://arxiv.org/abs/2501.00586

[8] Swathi, R., & Aruna, A. (2023). Using VGG16 algorithms for classification of lung cancer in CT scans image. *arXiv preprint*, arXiv:2305.18367. https://arxiv.org/abs/2305.18367

[9] Ahmed, M. M., & Al-Garni, A. Z. (2021). Enhancing lung cancer detection from lung CT scan using image processing and deep neural networks. *Revue d'Intelligence Artificielle*, *37*(6), 791–797. https://doi.org/10.18280/ria.370624

[10] Prakash, P., & Singh, A. (2022). VGG-19 with LSTM for lung cancer detection using CT images. *Journal of Biomedical Informatics*, *132*, 104102. https://doi.org/10.1016/j.jbi.2022.104102