

A project report on

CUSTOMIZED CONVOLUTIONAL NEURAL NETWORKS FOR PRECISE LUNG CANCER CLASSIFICATION

Submitted in partial fulfilment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

SIMHADRI MOHANA KUSHAL – 20BCE1952



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

**SCHOOL OF COMPUTER SCIENCE AND
ENGINEERING**

April, 2024

CUSTOMIZED CONVOLUTIONAL NEURAL NETWORKS FOR PRECISE LUNG CANCER CLASSIFICATION

Submitted in partial fulfilment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

SIMHADRI MOHANA KUSHAL – 20BCE1952



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

**SCHOOL OF COMPUTER SCIENCE AND
ENGINEERING**

April, 2024



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

DECLARATION

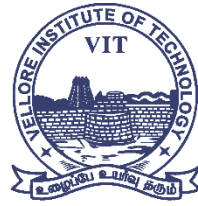
I hereby declare that the thesis entitled “Customized Convolution Neural Networks for precise Lung Cancer Classification” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of bonafide work carried out by me under the supervision of Dr Ilavarasi A K.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date:

Signature of the Candidate



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled “Customized Convolution Neural Networks for precise Lung Cancer Classification” is prepared and submitted by Simhadri Mohana Kushal(20BCE1952) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering programme is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

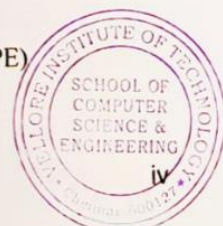
Ilavarasi
Signature of the Guide:
Name: Dr Ilavarasi A K
Date: 24/4/24

Kishore Ram
Signature of the External Examiner:
Name: KISHORE RAM TJ
Sr SW Engineer, Zeller
Date: 25/Apr/24

Dr. Gnanapriya
Signature of the Internal Examiner:
Name: Dr. Gnanapriya G S
Date: 25/04/2024

Approved by the Head of Department,
BTech CSE, *Nithyanandam*
Name: Dr. Nithyanandam P
Date:

(Seal of SCOPE)



ABSTRACT

This research focuses on the development and assessment of Customized Convolutional Neural Networks (CNNs) for enhancing lung cancer classification, a critical endeavour given the disease's complexity and the diversity of its subtypes. Traditional diagnostic methodologies often struggle to accurately differentiate between these subtypes, which is crucial for administering targeted treatments. By engineering CNN architectures specifically designed to detect subtle features characteristic of distinct lung cancer subtypes, this study seeks to surpass the diagnostic capabilities of standard models and pre-trained networks. The investigation includes a thorough review of related works, highlighting the advancements in pre-trained models for medical imaging and their adaptation for lung cancer classification. It introduces a customized CNN model, meticulously developed to leverage unique patterns in lung cancer imaging, thereby aiming to improve sensitivity and diagnostic accuracy. Methodologically, the study encompasses steps for data preprocessing, feature extraction using Vision Transformers (ViTs), and dimensionality reduction, culminating in a robust analysis aimed at classifying lung cancer more accurately. Comparative evaluations between the custom CNN model and pre-trained models reveal that the former not only exhibits superior performance in detecting nuanced patterns indicative of lung cancer but also achieves this with remarkable training efficiency. Conclusively, the research underscores the significance of customizing CNN architectures for lung cancer diagnostics. It demonstrates the potential of tailored deep learning models to significantly enhance the precision and efficiency of lung cancer classification, supporting the integration of advanced AI-driven diagnostic methods into clinical settings for improved patient care and treatment outcomes.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr Ilavarasi A K, Assistant Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Healthcare Analytics.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, B.Tech. CSE and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date:

Simhadri Mohana Kushal

CONTENTS

| | |
|-------------------------------|------------|
| CONTENTS | vii |
| LIST OF FIGURES | ix |
| LIST OF ACRONYMS | x |

CHAPTER 1

INTRODUCTION

| | |
|-----------------------------------|---|
| 1.1 INTRODUCTION | 1 |
| 1.2 CHALLENGES IN DIAGNOSIS | 2 |
| 1.3 PROJECT PROPOSITION | 3 |
| 1.4 SCOPE OF THE PROJECT | 5 |

CHAPTER 2

BACKGROUND

| | |
|--|----|
| 2.1 BACKGROUND | 7 |
| 2.2 CHALLENGES OF LUNG CANCER | 9 |
| 2.3 EVOLUTION OF DIAGNOSTIC TECHNOLOGIES | 10 |
| 2.4 RISE OF AI IN HEALTHCARE | 11 |
| 2.5 CUSTOMIZED CNN FOR LUNG CANCER DIAGNOSIS | 12 |
| 2.6 POTENTIAL IMPACT & FUTURE DIRECTIONS | 13 |
| 2.7 RELATED WORKS | 14 |

CHAPTER 3

METHODOLOGY

| | |
|--|----|
| 2.1 METHODOLOGY | 17 |
| 2.2 LOADING DATA & PRE-PROCESSING | 18 |
| 2.3 FEATURE EXTRACTION USING VISION TRANSFORMERS ... | 20 |
| 2.4 APPLYING PRE-TRAINED MODEL | 22 |

| | | |
|-------|-------------------------------|----|
| 2.3.1 | RESNET MODEL | 22 |
| 2.3.2 | VGGNET MODEL | 23 |
| 2.3.3 | GOOGLENET MODEL | 23 |
| 2.3.4 | DENSENET MODEL | 25 |
| 2.5 | CUSTOM CNN MODEL | 25 |
| 2.3.1 | ARCHITECTURE | 25 |
| 2.3.2 | TRAINING PROCESS | 30 |
| 2.6 | HYPER PARAMETER TUNING | 31 |
| 2.7 | K-FOLD CROSS VALIDATION | 33 |

CHAPTER 4

CONCLUSION AND FUTURE WORKS

| | | |
|-----|------------------------|----|
| 2.1 | RESULTS OBTAINED | 34 |
| 2.2 | CONCLUSION | 37 |
| 2.3 | FUTURE WORKS | 38 |

CHAPTER 5

REFERENCES

| | | |
|-----|------------------|----|
| 2.1 | REFERENCES | 40 |
|-----|------------------|----|

APPENDIX

| | | |
|----|------------|----|
| 1. | CODE | 44 |
|----|------------|----|

List of Figures

| Figure Number | Page Number |
|--|-------------|
| Figure 1 – Workflow Diagram | 17 |
| Figure 2 – Lung Aadenocarcinomas | 18 |
| Figure 3 – Benign Lung | 19 |
| Figure 4 – Lung Squamous Cell Carcinomas | 19 |
| Figure 5 – Extracted Feature Graph | 21 |
| Figure 6 – Custom CNN model Flowchart | 25 |
| Figure 7 – Hyper Parameter Results | 32 |
| Figure 8 – K-cross Validation Results | 33 |
| Figure 9 – VGG Model Results | 35 |
| Figure 10 – GoogleNet Model Results | 35 |
| Figure 11 – ResNet Model Results | 35 |
| Figure 12 – DenseNet Model Results | 36 |
| Figure 13 – Custom CNN Model Results | 36 |

List of Acronyms

| Acronyms | Meaning |
|----------|-------------------------------|
| CNN | Convolutional Neural Networks |
| CT | Computed tomography |
| MRI | Magnetic Resonance Imaging |
| AI | Artificial Intelligence |
| ViT | Vision Transformers |

CHAPTER 1

INTRODUCTION

Lung cancer, known for its high mortality rate and significant heterogeneity, poses a major challenge in the field of oncology, calling for a move towards more advanced diagnostic techniques to improve treatment effectiveness and patient outcomes. Traditional diagnostic methods, although fundamental, often fail to accurately categorize lung cancer subtypes, which is critical for influencing therapeutic decisions and prognosis [1]. Advances in medical imaging technology have greatly aided in obtaining accurate visualizations and a thorough understanding of lung cancer's complex manifestations. However, interpretations of the data collected by these advanced devices still face challenges and errors, often resulting in diagnostic inaccuracies. In contrast, artificial intelligence, particularly Customized Convolutional Neural Networks (CNNs), stands out as a beacon of hope for developing sophisticated methods to improve lung cancer classification [6]. These AI models, adapted to distinguish the subtle differences among lung cancer subtypes, could greatly enhance diagnostic accuracy beyond what is possible with conventional methods or generic AI models. The goal is to design and optimize a CNN architecture that leverages recent advances in deep learning for lung cancer imaging, thereby addressing the current diagnostic gaps and the need for precision medicine. Custom CNNs represent a cutting-edge approach for grading histopathological images, paving the way for advancements in personalized medicine and improved patient outcomes[7]. This study aims to provide enlightening insights to the diagnostic community, ushering in a new era of precision diagnostics in oncology through AI. With earlier detection and precise classification by custom-made CNNs, this approach could significantly reduce the burden of lung cancer for patients worldwide.

CHALLENGES IN DIAGNOSIS

Diagnosis of lung cancer presents a complex challenge due to the disease's heterogeneity, subtle manifestations, and the limitations of conventional diagnostic methods. These traditional approaches, despite being fundamental to medical practice, often lack the precision needed to differentiate between lung cancer's many histological subtypes, crucial for determining effective treatment pathways. This gap significantly affects treatment, patient management, and ultimately, survival outcomes[1]. The diagnostic dilemma is exacerbated by the limitations of imaging technologies and biopsy procedures. Despite advancements in resolution and clarity, imaging relies heavily on the interpretative expertise of radiologists to identify often indistinct patterns denoting different subtypes. Biopsies, though definitive, carry risks due to their invasive nature. The need for diagnostic tools that are both accurate and adaptable to the dynamic evolution of lung cancer underscores a critical gap in clinical management, necessitating innovative diagnostic methodologies for more precise and effective disease management [2].

The advent of artificial intelligence, especially deep learning models like convolutional neural networks (CNNs), promises a breakthrough in overcoming these obstacles, offering unprecedented diagnostic accuracy and efficiency. However, integrating AI into lung cancer diagnostics faces challenges, including developing models capable of accurately interpreting complex imaging data and addressing ethical concerns about AI in healthcare. Moreover, deploying AI-driven diagnostic tools in clinical settings raises issues regarding acceptance by healthcare professionals and integration into existing workflows [5]. Despite these challenges, the pursuit of improved diagnostic solutions for lung cancer continues, driven by the goal to enhance early detection, enable accurate subtype classification, and facilitate personalized treatment approaches that significantly improve patient outcomes against this formidable disease.

PROJECT PROPOSITION

The proposal to use custom convolutional neural networks (CNNs) to improve diagnostic accuracy in lung cancer faces a complex array of both technical and conceptual challenges. These challenges range from designing AI models finely tuned to subtly differentiate lung cancer subtypes, to dealing with the variability inherent in the disease's presentation. This endeavor carries the promise of a paradigm shift towards precision medicine, requiring a carefully calibrated CNN architecture to accurately interpret complex patterns in medical imaging. This task involves collecting and curating large and diverse datasets that capture the full spectrum of lung cancer manifestations, a process often hindered by issues related to data privacy, availability, and the representativeness of disease heterogeneity.

The personalized nature of these models demands a tight integration of oncological expertise with advanced computational techniques, bridging clinical insights with AI capabilities—a synergy that is difficult to achieve but essential for the models' effectiveness. Furthermore, the dynamic nature of lung cancer, with its potential for rapid progression and mutation, adds another layer of complexity. AI systems must not only accurately classify current conditions but also be adaptable, learning from new data to stay relevant as the disease landscape evolves. Implementing such systems involves significant technical challenges, including ensuring model stability, preventing data drift, maintaining patient privacy, and adhering to regulatory standards [3].

This endeavor also encounters skepticism within the medical community regarding the integration of AI-driven diagnostic tools into established clinical workflows, presenting substantial cultural and logistical challenges. Overcoming these hurdles requires clear evidence of the technology's efficacy, reliability, and safety to gain acceptance among healthcare professionals. Moreover, the journey from research prototypes to clinically validated tools involves rigorous testing and regulatory approval processes, posing significant barriers to the rapid deployment of these innovations in healthcare.

Despite these challenges, the project remains a beacon of hope for transforming lung cancer diagnostics, embodying the convergence of oncology and artificial intelligence. It aims to usher in a new era of accurate, efficient, and personalized cancer care, assuming these multifaceted hurdles can be successfully navigated.

SCOPE OF RESEARCH

Exploring the use of Customized Convolutional Neural Networks (CNNs) for lung cancer diagnosis entails navigating through a complex and multifaceted landscape, bridging gaps between the nuanced manifestations of the disease and the cutting-edge realms of AI technology. This journey demands not only a deep understanding of lung cancer's biological and pathological aspects but also a thorough command over the technological and computational frameworks needed to develop advanced diagnostic tools. The challenge spans from technical obstacles, such as designing and training AI models to detect subtle differences between lung cancer types, to logistical barriers in integrating these innovations into clinical practice and healthcare systems.

A critical hurdle is obtaining high-quality, diverse, and well-annotated datasets that accurately represent the wide range of lung cancer cases. These datasets are vital for training AI models capable of generalizing their diagnostic capabilities across the global patient population. Furthermore, the goal to surpass the diagnostic precision and speed of existing methodologies sets a high bar, necessitating technological innovation and methodological rigor to convincingly demonstrate improvements through comparative analysis and validation studies.

Moreover, the introduction of AI-driven diagnostic tools in healthcare necessitates making these technologies accessible and interpretable to medical professionals, calling for advancements in user interface design, model explainability, and the integration of AI insights into clinical decision-making processes. Ethical considerations, such as patient privacy, data security, and the mitigation of algorithmic bias, must be meticulously navigated to ensure the responsible application of AI in healthcare [3].

The scope of research also encompasses the dynamic and evolving nature of lung cancer itself, requiring AI models to be adaptable and scalable to incorporate new information as our understanding of the disease progresses. This demands ongoing development and maintenance efforts to keep the models current and effective [3].

The ambition to significantly contribute to the domain of AI-enabled healthcare innovations through the development of customized CNNs for lung cancer diagnosis represents a visionary approach that introduces a complex array of challenges. Each challenge must be addressed with precision, creativity, and a collaborative spirit that merges the domains of oncology, computational science, and clinical practice, aiming to transform lung cancer diagnostics through the convergence of these fields[5].

CHAPTER 2

BACKGROUND

This section outlines the intricate journey of lung cancer diagnostics, highlighting the shift from initial challenges to the promising advancements brought about by artificial intelligence (AI). The complexity of lung cancer and the limitations of conventional diagnostic methods have long hindered the fight against this leading cause of cancer-related deaths worldwide[4]. The need for precise classification of lung cancer's diverse histological subtypes to determine effective treatment has fueled technological evolution in diagnostics, moving from basic X-ray imaging to more sophisticated techniques like CT scans, MRI, and PET scans. Despite these advancements providing deeper insights into the disease, interpreting detailed imaging data requires high expertise and is prone to human error.

The advent of AI, particularly deep learning and Convolutional Neural Networks (CNNs), offers a new hope, promising to overcome these longstanding challenges. Celebrated for their image recognition capabilities, CNNs aim to reveal patterns in medical imaging that may escape human detection[5]. This research focuses on customizing CNN architectures to tackle the specific challenges of lung cancer diagnosis, aiming to develop a diagnostic tool that outperforms traditional methods and generic AI models by identifying the subtle differences between lung cancer subtypes.

This endeavor to create customized CNNs for lung cancer diagnosis is not just a testament to technological innovation but also a significant step towards precision medicine[8]. The potential of these AI models to improve diagnostic accuracy heralds transformative outcomes for patients, facilitating earlier detection and precise classification of lung cancer subtypes. Furthermore, exploring these specialized neural networks not only aims to enhance lung cancer diagnostics but also lays the foundation for future AI-driven healthcare solutions. The project's impact may extend beyond oncology, embedding AI into diagnostics across various medical fields, advancing personalized medicine, and broadening the possibilities for patient care and treatment outcomes. Rooted in historical challenges and technological evolution, this project

leads the way in shifting healthcare paradigms, redefining cancer diagnosis and treatment through innovative AI applications.

CHALLENGES OF LUNG CANCER

The difficulties in diagnosing lung cancer encompass a multifaceted dilemma, primarily due to the disease's aggressive nature and the wide spectrum of histological subtypes. These aspects position lung cancer as the leading cause of cancer-related mortality globally. The complexities in diagnosis begin with the inherent challenge in accurately identifying subtypes, a critical step for tailoring effective treatment plans. Traditional diagnostic methods often fall short in sensitivity and specificity, highlighting an urgent need for more sophisticated solutions capable of precisely navigating the nuanced landscape of lung cancer. The dynamic nature of the disease, which can evolve or present in various forms, demands diagnostic tools that are both accurate and adaptable to the shifting paradigms of oncological pathology[7].

Reliance on conventional diagnostic techniques, such as biopsies and imaging tests, introduces additional complexities. These methods can be invasive and their data challenging to interpret, often depending on high levels of expertise and susceptible to human error. Furthermore, lung cancer's classification into multiple subtypes, each with distinct genetic, molecular, and clinical characteristics, adds another layer of diagnostic challenge. This scenario is made more critical by the timing of lung cancer diagnosis, where delays or inaccuracies can significantly impact patient prognosis and survival rates.

These challenges not only affect the efficacy of lung cancer diagnosis and treatment but also highlight the gaps in existing healthcare systems to adequately address the complexities of this formidable disease. Consequently, there is a significant emphasis on developing innovative diagnostic tools and methods capable of overcoming these hurdles. Advances in medical technology and the potential of artificial intelligence are key to revolutionizing lung cancer diagnosis, offering hope for improved patient outcomes against a disease that continues to present a significant global health challenge.

EVOLUTION OF DIAGNOSTIC TECHNOLOGY

The evolution of diagnostic technology, particularly in the context of lung cancer, represents a journey marked by groundbreaking advancements and significant challenges, shaping the way healthcare professionals approach one of the most formidable diseases in oncology. From the early days of reliance on basic X-ray imaging, which provided limited insights into the disease's manifestations, to the sophisticated realms of Computerized Tomography (CT) scans, Magnetic Resonance Imaging (MRI), and Positron Emission Tomography (PET) scans, each technological leap has offered deeper, more detailed windows into the body's internal state. Yet, these advancements come with their own set of challenges, notably the complexity of interpreting the rich data these technologies produce, a task that demands a high level of expertise and remains prone to human error, reflecting the nuanced intricacies of lung cancer's presentation[3]. Furthermore, the rapid pace of technological innovation, while promising, introduces issues of accessibility and standardization across healthcare systems, with cutting-edge diagnostic tools often accompanied by high costs and a need for specialized training, potentially limiting their widespread adoption and leading to disparities in patient care. Additionally, the integration of these technologies into clinical workflows poses logistical challenges, requiring not just the physical infrastructure to support their use but also the development of protocols that balance the benefits of advanced diagnostics with the risks and discomforts associated with more invasive procedures[5]. Moreover, as diagnostic technologies advance, they push the boundaries of what is known about lung cancer, uncovering new subtypes and complexities of the disease, which, in turn, demand further refinement of diagnostic tools and methodologies, creating a continuous cycle of innovation and adaptation. This environment of constant evolution, while driving progress in lung cancer diagnosis, necessitates a collaborative approach among researchers, clinicians, and technologists to overcome the hurdles of implementation, ensure the ethical use of advanced diagnostics, and ultimately improve patient outcomes. The challenge, therefore, lies not just in developing new diagnostic technologies but also in ensuring they are accessible, interpretable, and effectively integrated into the multifaceted landscape of lung cancer care, a task that underscores the ongoing quest for precision medicine and the pivotal role of innovation in shaping the future of oncology diagnostics[1].

RISE OF AI IN HEALTHCARE

The rise of artificial intelligence (AI) in healthcare, heralding transformative potential across diagnostics, treatment planning, and patient care, confronts a spectrum of challenges that underscore the complexity of its integration into the medical field. At the forefront, data privacy and security concerns loom large, as the deployment of AI-driven solutions necessitates access to vast amounts of sensitive patient data, posing risks of breaches and unauthorized access that could undermine patient trust and compliance with strict regulatory frameworks like HIPAA[4]. Furthermore, the accuracy and reliability of AI systems, while promising, are contingent upon the quality and diversity of the training data, which often reflects inherent biases or lacks representation across demographics, potentially leading to skewed or erroneous outcomes that could exacerbate disparities in healthcare. The interpretability of AI decisions remains a critical hurdle, with the so-called "black box" nature of deep learning models making it challenging for clinicians to understand the rationale behind specific recommendations, thereby hindering their adoption and integration into clinical decision-making processes. Additionally, the current healthcare workforce is largely unprepared for the widescale adoption of AI, with a significant gap in AI literacy and a need for extensive training and education to enable healthcare professionals to effectively leverage these technologies. Technical limitations and the computational cost of developing, training, and implementing AI models also present significant barriers, especially in resource-constrained settings, potentially limiting the accessibility and scalability of AI solutions across the healthcare ecosystem. Ethical considerations, including concerns about automating care and the potential for AI to recommend treatments that may conflict with patient or clinician preferences, further complicate the landscape. Moreover, regulatory challenges slow the pace of AI adoption, with existing frameworks often ill-equipped to evaluate the safety and efficacy of AI applications in healthcare, necessitating new guidelines and standards to ensure patient safety and the responsible use of AI[4]. The amalgamation of these challenges paints a complex picture of the journey toward the integration of AI in healthcare, a path fraught with technical, ethical, and logistical obstacles that must be navigated with caution to fully realize the potential of AI to revolutionize medical care and outcomes.

CUSTOMIZED CNN FOR LUNG CANCER DIAGNOSIS

The endeavour to implement Customized Convolutional Neural Networks (CNNs) for lung cancer diagnosis confronts a myriad of challenges, encompassing technical intricacies, data limitations, and clinical integration hurdles[5]. At the forefront lies the imperative to develop CNN architectures finely tuned to discern the subtle nuances among lung cancer subtypes, necessitating a deep understanding of both oncological pathology and advanced computational techniques. Acquiring sufficiently diverse and annotated datasets representative of the full spectrum of lung cancer manifestations poses a formidable obstacle, compounded by issues of data privacy, availability, and reliability[9]. Moreover, the customization process demands meticulous attention to model optimization and validation, ensuring robust performance and generalizability across patient populations. Integration into clinical workflows presents another layer of complexity, requiring seamless interoperability with existing diagnostic practices and garnering acceptance from healthcare professionals[6]. Ethical considerations, including patient privacy, data security, and algorithmic bias, further underscore the need for responsible AI deployment in healthcare settings. Despite these challenges, the pursuit of customized CNNs for lung cancer diagnosis heralds a transformative approach to precision medicine, with the potential to significantly improve diagnostic accuracy and patient outcomes in the fight against this formidable disease.

POTENTIAL IMPACT AND FUTURE DIRECTION

The realization of the potential impact and future direction of customized Convolutional Neural Networks (CNNs) in lung cancer diagnosis faces a multitude of challenges and considerations. Foremost among these challenges is the translation of research findings into meaningful clinical applications, necessitating rigorous validation studies, regulatory approvals, and seamless integration into existing healthcare workflows. Achieving widespread adoption of AI-driven diagnostic tools also requires addressing concerns regarding data privacy, security, and algorithmic transparency, while ensuring equitable access and minimizing potential biases in healthcare delivery. Furthermore, the scalability and sustainability of AI-powered solutions pose significant logistical and infrastructural challenges, particularly in resource-constrained healthcare settings. Beyond these immediate hurdles, the long-term impact of customized CNNs in lung cancer diagnosis hinges on ongoing research and development efforts aimed at enhancing model performance, expanding diagnostic capabilities to encompass emerging subtypes and evolving disease characteristics, and fostering interdisciplinary collaborations to drive innovation in AI-driven healthcare. Ultimately, the successful realization of the potential impact and future direction of customized CNNs in lung cancer diagnosis will require a concerted effort from stakeholders across the healthcare ecosystem, guided by a shared commitment to leveraging technology to improve patient outcomes and advance the practice of precision medicine in oncology.

RELATED WORKS

To forge a custom CNN model, an exploration into the utility of pre-trained models for lung cancer classification and its variants is crucial. Sun et al. (2022) [10] unveiled an innovative algorithm for lung cancer image classification and segmentation, harnessing the capabilities of an enhanced Swin Transformer. This method represents a significant step forward by adapting a transformer model, originally prominent in natural language processing, for medical imaging tasks in computer vision, achieving classification accuracies up to 82.3% and segmentation accuracies over 95%.

In 2023, Van der Burgh et al. [11] utilized pre-trained language models based on the RoBERTa framework for predicting lung cancer using unstructured Dutch medical records from primary healthcare settings. Their investigation into soft-prompt tuning as a novel adaptation strategy for these pre-trained models revealed its superiority over conventional fine-tuning techniques, emphasizing the adaptability of pre-trained models to address challenges associated with skewed class distributions and improve prediction accuracy.

Baranwal et al. [12] (2021) focused on leveraging Convolutional Neural Networks (CNNs) for the classification of lung cancer histopathology images, employing renowned architectures such as ResNet 50, VGG-19, Inception_ResNet_V2, and DenseNet for extracting features. Their research highlighted the effectiveness of pre-trained networks in distinguishing between normal, adenocarcinoma, and squamous cell carcinoma categories, using triplet loss to fine-tune the models for enhanced separation between clusters.

Moreover, Alzubaidi et al. [13] (2021) introduced MedNet, a pre-trained CNN model specially designed for medical imaging applications. Faced with the challenge of limited data availability for training deep learning models in medical imaging, MedNet employs transfer learning from a broad spectrum of medical images to significantly improve the model's generalization capability across various medical imaging tasks, including lung cancer classification.

In the realm of early lung cancer detection through CT scans, Mamun et al. [14] (2023) developed the LCDctCNN, a CNN-based model that outperformed established models

like Inception V3, Xception, and ResNet-50 in terms of accuracy, Area Under the Curve (AUC), recall rate, and loss metrics.

Pandit et al. [15] (2022) revealed a neural network for lung cancer classification that incorporates an optimization function, enhancing prediction accuracy and reducing processing times through the innovative use of an autoencoder and multi-space imaging techniques in the pooling layer.

Li et al. [16] (2018) proposed an algorithm that integrates handcrafted features with deep CNN outputs for predicting the malignancy of lung nodules, demonstrating the benefits of combining diverse data types for improved diagnostic accuracy.

Continuing this line of research, Bhoj Raj Pandit et al. [17] emphasized a deep learning neural network optimized for lung cancer classification, employing an autoencoder and multi-space imaging to achieve significant advancements in diagnostic accuracy and efficiency.

Junhua Chen et al. [18] introduced a computer-aided diagnosis (CAD) system for lung cancer, approaching the diagnostic process as a multiple instance learning (MIL) challenge and leveraging radiomics as input features. Their method utilizes a deep attention-based MIL classifier to enhance diagnostic accuracy and interpretability.

Hasan Hejbari Zargar et al. [19] utilized the VGG16 deep learning algorithm for classifying lung nodules from CT scans, demonstrating the model's effectiveness in assisting medical professionals in accurately diagnosing lung cancer nodules.

"Lung-Net: A Deep Learning Framework for Lung Tissue Segmentation in Three-Dimensional Thoracic CT Images" [20] showcases Lung-Net, an advanced deep learning framework for segmenting lung tissue in 3D CT images, achieving remarkable accuracy across several datasets.

Shimazaki et al. [21] detailed the development of a deep learning model for lung cancer detection in chest radiographs, addressing the challenges of detecting cancers in radiographically dense areas and enhancing lung cancer detection in chest imaging.

Lastly, the "Deep learning ensemble 2D CNN approach towards the detection of lung cancer" [22] leverages an innovative Deep Ensemble 2D CNN architecture for lung

nodule identification in CT scans, demonstrating the significant impact of deep learning on medical imaging diagnostics.

CHAPTER 3

METHODOLOGY

Customized Convolutional Neural Networks (CNNs) models are developed around our research, for lung cancer classification specifically identifying. The difficulty of early detection and classification of lung cancer is recognized, and our system is focused on the neural networks architectures being optimized to capture subtle features indicative of specific histological types.

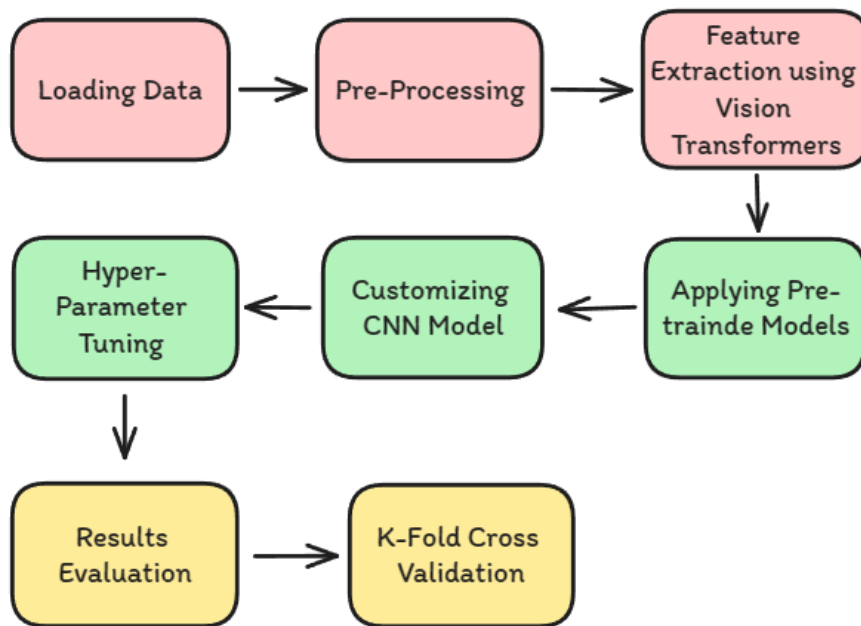


Figure 1 - Workflow Diagram

LOADING DATA AND PRE-PROCESSING

Only the Lung cancer images from LC25000 dataset are loaded. A re-sizing function is used while loading the dataset to keep the image size consistent to 224x224 pixels, which is a common practice in deep learning. The image is then converted into tensors with the help of PyTorch library, which would be our input for the deep learning models. The image is normalized to scale the image pixels between 0 and 1. This helps in the effective training of our deep learning models.

For pre-processing the image, the image is first converted into LAB color space from RGB color space. This helps in getting an intensity focused information channel, which contains relevant information for lung cancer classification. Binary thresholding is then applied to segment the image and create a binary mask. This is useful in isolating regions of interest or specific features in the image, aiding in the detection of patterns related to lung cancer.

Morphological operations are used to clean and refine the binary mask to smoothen and close small gaps or holes in the mask, which can improve the quality of the segmented regions. The binary mask is now applied to the original RGB image, resulting in a pre-processed image. This image is then saved for further analysis. These series of processes result in an image where all the regions of interest are emphasised, helping in our objective of lung cancer classification.

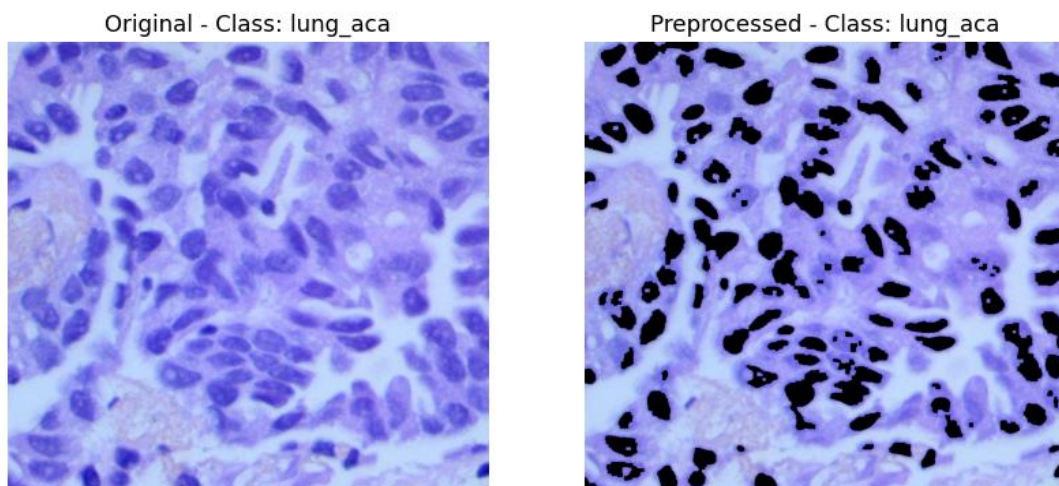


Figure 2 - Lung Aadenocarcinomas

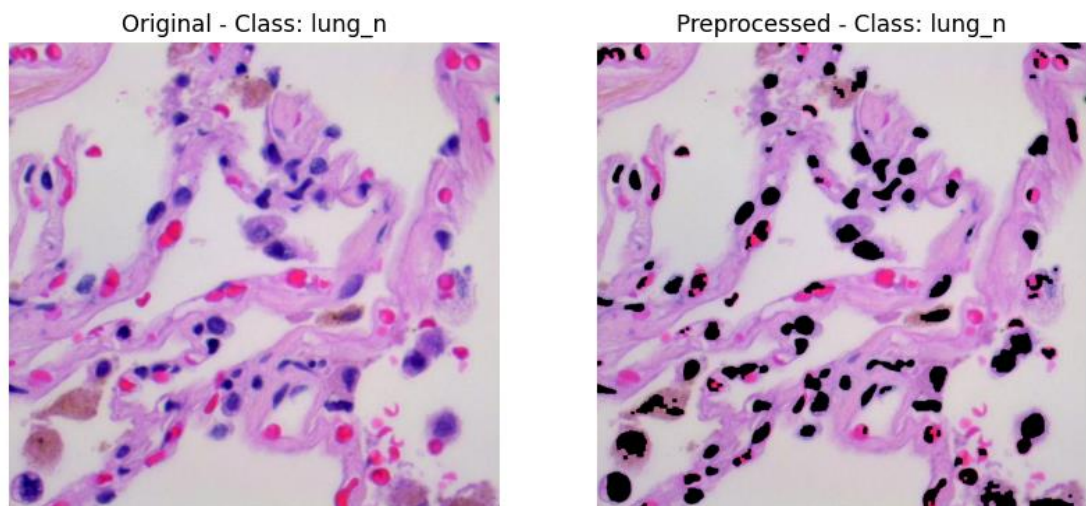


Figure 3 - Benign Lung

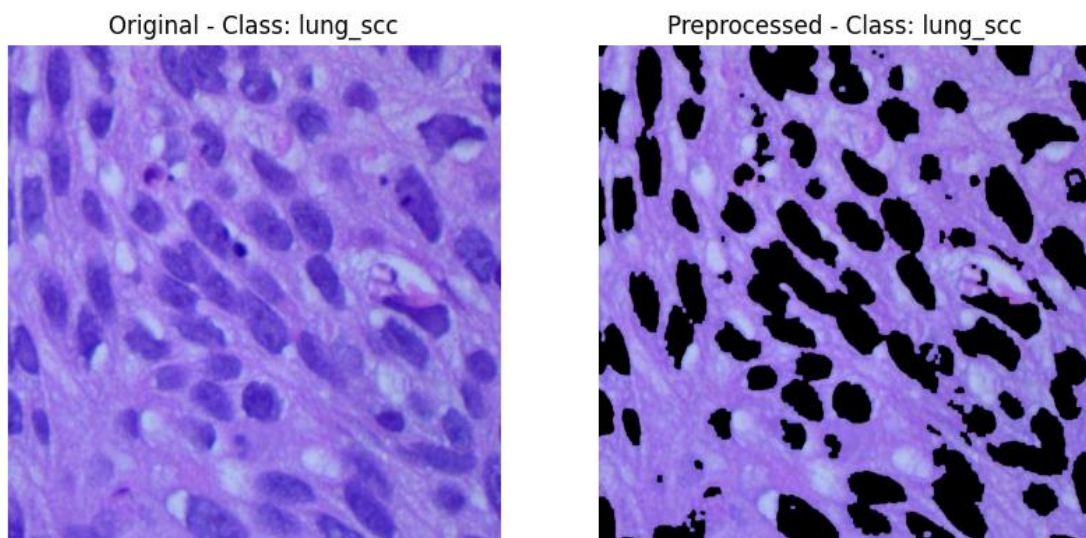


Figure 4 - Lung Squamous Cell Carcinomas

FEATURE EXTRACTION USING VISION TRANSFORMERS

Vision Transformers (ViTs) have emerged as a powerful tool for feature extraction in medical imaging by leveraging a self-attention mechanism inherent to the Transformer architecture. In this methodology, input images are segmented into fixed-size patches, transformed into linear embeddings, and processed alongside positional embeddings through a Transformer encoder. This process enables the relevance of each segment in relation to the rest to be weighed by the ViT, capturing both local and global context within the image. Through the multi-head attention mechanism, features from various perspectives can be extracted by ViTs, enhancing their capability to learn complex patterns. This is particularly beneficial in medical domains such as lung cancer classification, where understanding intricate image details is crucial [23].

For lung cancer diagnosis, the recognition of long-range dependencies within medical images by ViTs is invaluable. Unlike traditional Convolutional Neural Networks (CNNs), in analyzing varied image structures and complex patterns, ViTs excel, essential for identifying subtle abnormalities in lung imagery. A deeper understanding of the relationships within the image is facilitated by the self-attention mechanism, aiding in the nuanced task of lung cancer classification. Enhanced performance and interpretability are led to by the improved feature extraction provided by ViTs, making them an ideal choice for addressing the complexities of lung cancer diagnostics.

Features with Vision Transformers are extracted upon, and for dimensionality reduction and visualization, t-Distributed Stochastic Neighbor Embedding (t-SNE) is employed, projecting the features into a two-dimensional space. The distribution and clustering of features related to different lung cancer types are elucidated by this visualization technique, offering insights into their separability and the discriminative power of the model's learned representations. Such visualizations are instrumental in understanding the efficacy of ViTs in distinguishing between various lung cancer subtypes, showcasing their potential in medical imaging applications.

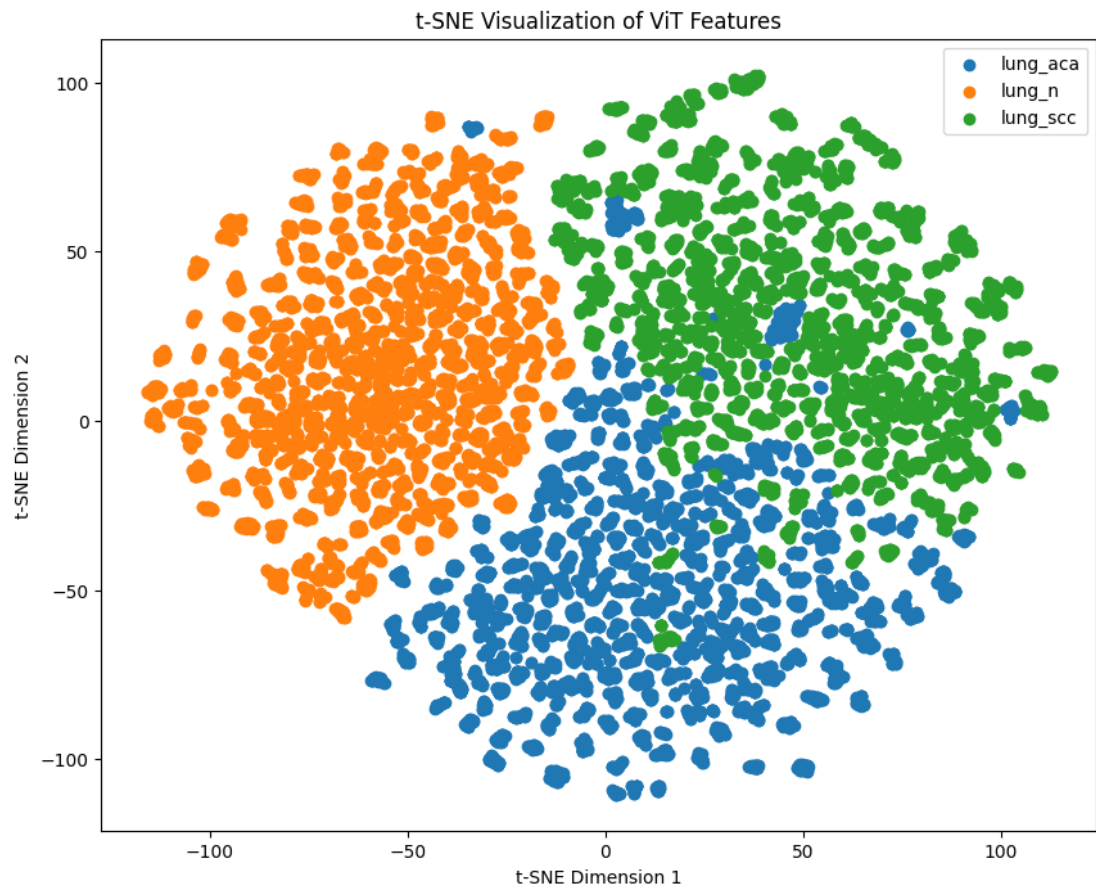


Figure 5 - Extracted Feature Graph

APPLYING PRE-TRAINED MODEL

ResNet Model

The Residual Neural Network (ResNet) framework has emerged as a pivotal innovation in the realm of medical image analysis, showcasing its effectiveness particularly in the classification of lung cancer from biopsy samples. ResNet introduces skip connections, a novel feature that tackles the vanishing gradient problem by establishing direct pathways for gradient propagation during the backpropagation phase. This innovation is crucial for the successful training of deep neural networks. Highlighting this approach, a recent study utilized ResNet to categorize lung cancer biopsy samples into distinct types, with the goal of elevating the accuracy of diagnoses and, consequently, enhancing patient care outcomes. ResNet's methodology incorporates an array of convolutional and pooling layers, culminating in fully connected layers dedicated to the task of classification. The integration of skip connections permits the network to adeptly learn residual functions relative to the inputs of layers, significantly boosting its ability to discern minute yet critical features indicative of various lung cancer types and their respective stages. This attribute renders ResNet exceptionally capable in differentiating among the diverse tissue patterns that characterize the spectrum of lung cancer, thereby amplifying the model's diagnostic accuracy [24].

VGGNet Model

The VGGNet architecture, conceived by Simonyan and Zisserman in 2014, has emerged as a fundamental structure in the realm of deep convolutional neural networks, acclaimed for its simplicity and exceptional performance across a variety of computer vision tasks, including the nuanced field of medical imaging. Characterized by its series of convolutional layers, each employing compact 3x3 filters, followed by max-pooling layers arranged into distinct convolutional blocks, VGGNet distinguishes itself through its depth. The architecture is available in different variants, notably VGG16 and VGG19, which are composed of 16 and 19 layers, respectively. Despite its straightforward design, VGGNet has played a crucial role in propelling forward computer vision applications by facilitating the extraction of rich, layered visual data representations.

Within the specific application of lung cancer classification via tissue images, VGGNet has shown exceptional potential. The architecture's deep convolutional layers are instrumental in unraveling hierarchical features critical for the precise categorization of lung tissue types associated with various cancer stages. This depth enables the network to capture both multi-scale and abstract features, which are indispensable for discerning subtle distinctions in tissue architectures, thereby effectively differentiating between malignant and benign conditions. Furthermore, VGGNet's architecture, with its extensive receptive fields within the convolutional layers, is pivotal for the detection of spatial irregularities in lung tissue images, underscoring its significance in enhancing the accuracy of medical diagnostic processes [25].

GoogLeNet Model

In 2014, Szegedy and his team introduced GoogLeNet, or Inception v1, heralding a significant leap in the evolution of deep convolutional neural networks (CNNs) through the integration of inception modules. These modules are characterized by their parallel convolutional layers with varying kernel sizes, allowing the network to adeptly capture and analyze features at multiple spatial resolutions. This groundbreaking approach not only elevated the network's overall performance but also enhanced its computational efficiency.

Within the realm of lung cancer diagnosis through the analysis of tissue images, GoogLeNet has demonstrated remarkable potential. By utilizing its inception modules to concurrently process features at different scales, GoogLeNet excels in detecting both localized and diffuse patterns indicative of lung cancer. The architecture's ability to discern complex interrelations among features in tissue images substantially boosts its accuracy in distinguishing malignant from benign tissues. Hence, GoogLeNet stands out as a crucial tool for the early detection and accurate classification of lung cancer, emphasizing its role in refining diagnostic precision and offering insightful analyses of lung tissue imagery, as suggested by recent research [26].

DenseNet Model

Introduced by Huang and colleagues in 2017, DenseNet stands out in the arena of deep learning architectures due to its novel approach of dense connectivity among layers. This design principle facilitates a direct linkage between each layer and every other subsequent layer, fostering an environment where feature reuse is optimized, gradient flow is enhanced, and feature propagation is robust across the network. The structure of DenseNet is characterized by its dense blocks—clusters of closely interconnected convolutional layers, interspersed with transition layers that serve to reduce dimensionality and consolidate feature maps.

In the realm of discerning lung cancer types through the analysis of tissue images, DenseNet has proven to be extraordinarily effective. Its hallmark dense connectivity excels in delineating complex spatial relationships and subtle features within tissue samples, which are critical indicators of lung cancer. The architecture's proficiency in maintaining and leveraging relevant feature information across layers leads to the extraction of highly distinct features. This capability significantly bolsters the accuracy in differentiating malignant from benign tissues, showcasing DenseNet's pivotal role in enhancing diagnostic precision in the study of lung cancer [27].

CUSTOM CONVOLUTION NEURAL NETWORK MODEL

Architecture

A specially designed CNN architecture tailored to the specific challenges and demands of classifying lung cancer from medical images is employed in our research. The layers used in the research are as follows:

1. **Convolutional Layers:** Serving as the core of the model, these layers extract pertinent features from the input lung images using trainable filters or kernels. These kernels convolve across the image, identifying patterns and structures at varying levels of detail. Through this process, the model discerns key indicators of lung cancer, including tumour shape, texture, and spatial arrangements.
2. **Activation Functions:** Post-convolution, the model applies ReLU activation functions across elements to introduce non-linearity, effectively mitigating the vanishing gradient issue and enabling the learning of intricate, non-linear feature relationships. This enhances the model's ability to differentiate among various lung abnormalities.
3. **Pooling Layers:** Strategically placed max-pooling layers down-sample feature maps, simplifying their dimensionality while highlighting dominant features. This is achieved by dividing the feature maps into distinct regions and preserving only the maximum value from each, thus bolstering the model's focus on critical features, and lessening computational demands.
4. **Batch Normalization:** Following each convolution and activation, batch normalization layers are integrated to foster a more stable and swift training process. By normalizing layer activations batch-wise, this technique effectively counters internal covariate shift, lessening initial configuration dependency and promoting a more consistent training experience.
5. **Dropout:** To combat overfitting, dropout layers randomly inactivate a subset of neurons during training. This introduces model redundancy, discouraging reliance on specific features and fostering a more robust and generalized learning outcome, enhancing the model's performance on novel lung images.

6. **Adaptive Pooling:** An adaptive average pooling layer adjusts feature map dimensions dynamically, ensuring uniform output sizes regardless of varying input image dimensions. This layer's flexibility allows the model to efficiently process images of different resolutions, preparing the data for the final classification layers.
7. **Fully Connected Layers:** The architecture concludes with layers that compile and interpret the features extracted earlier, culminating in a set of class predictions. These predictions are refined through a softmax activation function, yielding probabilities across target categories, and facilitating the identification of lung cancer presence in the images.

In essence, this custom CNN architecture is intricately engineered to adeptly extract and analyse features from lung cancer images, utilizing a blend of convolutional, pooling, normalization, dropout, and fully connected layers to achieve nuanced representations and precise classification.

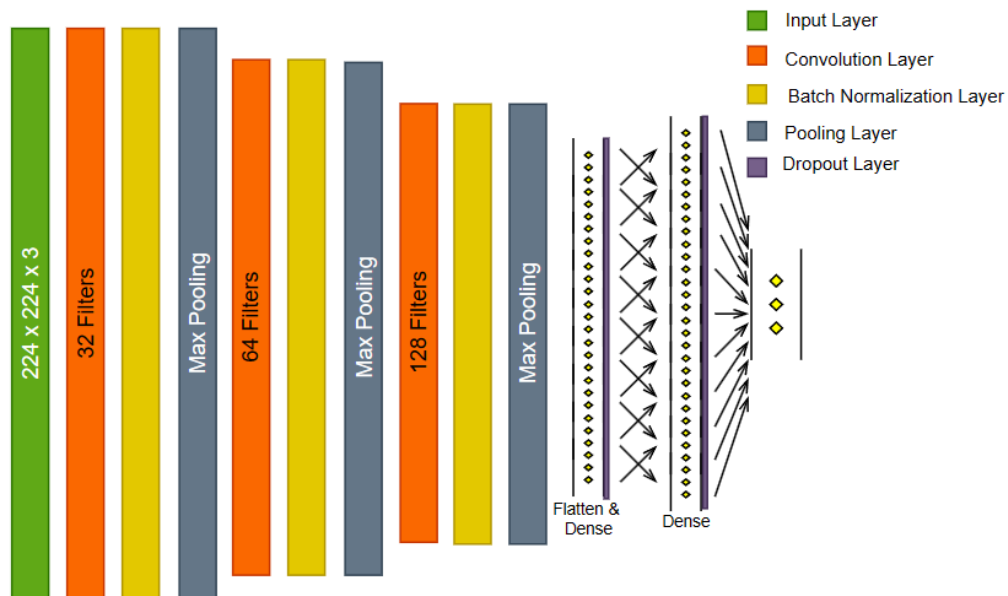


Figure 6 - Custom CNN Model Flowchart

Delving deeper into the customized CNN architecture utilized for classifying lung cancer from tissue images, let's explore each layer type and their specific roles within the network:

1. **Input Layer (224x224x3):** Initiating with an input layer that sets the dimensions of the input images at 224x224 pixels across 3 channels (RGB), this size strikes a balance between capturing sufficient detail and maintaining computational efficiency, a standard practice in deep learning model design.
2. **Conv2D (32 filters):** The network's inaugural convolutional layer employs 32 distinct filters to process the input image, each designed to detect various features such as edges, textures, or patterns. These convolutional layers are pivotal in CNNs, tasked with autonomously learning spatial hierarchies of features from the images.
3. **Batch Normalization:** This layer normalizes the convolutional layer's output, adjusting and scaling activations to stabilize the learning trajectory and diminish the training epochs necessary for deep network training.
4. **MaxPooling2D:** Pooling layers serve to down-sample the extracted image data, minimizing the spatial dimensions of the output volume. Max pooling selects the maximum value from each neuron cluster of the preceding layer, contributing to feature detection invariance regarding scale and orientation shifts.
5. **Conv2D_1 (64 filters):** Introducing a second convolutional layer with an increased filter count allows the network to discern more complex features, such as corners and edge combinations. An uptick in filters correlates with a deeper network, affording the learning of finer details.
6. **Batch Normalization_1:** Implementing batch normalization post-convolution enhances model construction, enabling quicker convergence and accommodating higher learning rates.
7. **MaxPooling2D_1:** Additional down-sampling streamlines the representation, easing the network's progression from spatial feature comprehension towards classification judgments.

8. **Conv2D_2 (128 filters):** Deeper in the network, the filter quantity escalates, permitting the learning of more sophisticated and abstract features.
9. **Batch Normalization_2:** This ensures the efficient training of even the network's deeper sections.
10. **MaxPooling2D_2:** This third max pooling phase further compacts the spatial representation, curtailing both the parameter count, and the computational effort required by the network.
11. **Global Average Pooling2:** Rather than transitioning feature maps to a dense layer via flattening, global average pooling averages each feature map directly, slashing the total parameter tally. This strategy is notably effective against overfitting.
12. **Flatten:** Post-global average pooling, the data is flattened into a vector to feed into dense layers. This process transforms the 2D feature matrix into a vector suitable for a fully connected neural network classifier.
13. **Dropout:** As a regularization technique, dropout layers during training randomly nullify a fraction of the layer's output features, aiding in overfitting mitigation.
14. **Dense (512 units):** A dense, or fully connected, layer follows, transforming the flattened output into a dimensional space where the network can make categorical decisions about the image.
15. **Dropout_1:** Employing a second dropout layer further combats overfitting, crucial for networks with extensive dense layers.
16. **Dense_1 (3 units):** The terminal dense layer condenses the learned feature dimensions to match the output class count, here three, possibly aligning with classifications like "non-cancerous," "cancerous non-small cell lung cancer," and "cancerous small cell lung cancer." Incorporating a softmax activation function here would allow for a probabilistic distribution over these classes.

In essence, this custom CNN architecture has been intricately engineered to adeptly extract and analyse features from lung cancer images, with a blend of convolutional,

pooling, normalization, dropout, and fully connected layers being utilized to achieve nuanced representations and precise classification.

Training process

The initialization of the model architecture marks the beginning of the training process for the customized CNN, encompassing the arrangement and configuration of convolutional layers, pooling layers, activation functions, batch normalization layers, dropout layers, adaptive pooling layers, and fully connected layers. Following this initiation, a loss function and optimizer are defined for the computation of the model's error and the update of its parameters, respectively. The dataset is subsequently split into training, validation, and test sets, with the training set being utilized iteratively to update the model's parameters. The training loop, which spans multiple epochs, involves the sequential processing of mini batches of data. A forward pass is conducted during each epoch, where predictions are generated, and the loss is computed via the chosen loss function. Gradients are then calculated through backpropagation, facilitating the update of parameters by the optimizer. The model's performance is assessed on the validation set after each epoch, leading to the informed adjustment of hyperparameters such as the learning rate and regularization parameters. Early stopping mechanisms are implemented to halt training if validation performance does not improve, thereby preventing overfitting. The trained model is finally evaluated on the test set to determine its generalization ability and performance on unseen data, thus providing insights into its efficacy for lung cancer classification tasks. Through this iterative process, the customized CNN is refined to effectively discern patterns indicative of lung cancer in medical images, thereby contributing to enhanced diagnostic accuracy and patient care.

HYPER PARAMETER TUNING

A meticulous iterative procedure focused on refining various parameters to boost the model's efficacy is entailed in optimizing the customized CNN for lung cancer classification. The magnitude of steps during gradient descent optimization is governed by a pivotal hyperparameter, the learning rate. Convergence may be hastened by a higher learning rate, yet it also poses the risk of surpassing the optimal minimum, while a lower learning rate might ensure stability but at the cost of slower convergence. Therefore, the evaluation of a spectrum of learning rates is crucial to finding an equilibrium between rapid convergence and stability. Additionally, adjustments are made to the batch size, which denotes the quantity of samples processed per training iteration. Training may be accelerated by larger batch sizes, but they might compromise the precision of gradient estimations, whereas smaller batch sizes tend to offer more precise gradient updates at the expense of convergence speed. Through trials with various values, the ideal batch size is determined to find an appropriate balance.

To mitigate overfitting—a prevalent challenge in deep learning models—regularization strategies are deemed essential. One approach to fine-tuning is adjusting the dropout rate, a method that intermittently omits a portion of neurons during training. Regularization is enhanced by elevating dropout rates, aiding in overfitting prevention but could detract from model performance if excessively high. Model complexity is restrained and overly large parameter values are averted by calibrating L2 regularization, which introduces a penalty based on the squared magnitude of model weights to the loss function.

The optimization extends to architectural hyperparameters as well, including the count of layers, sizes of filters, and dimensions of kernels. The model's capability to extract pertinent features from the input data is significantly affected by these aspects. The optimal setup that augments the model's precision in classifying lung images is identified through systematic alterations to these architectural elements and assessing their influence on performance metrics.

By engaging in thorough experimentation and hyperparameter fine-tuning, the customized CNN is fine-tuned to an ideal state that elevates its lung cancer classification performance. This detailed process ensures the model's robustness, superior generalization to new data, and its proficiency in detecting lung cancer indicators, thereby enhancing diagnostic precision and patient outcomes.

| Results Summary: | | | | | | | |
|------------------|---------------|------------|----------|-----------|----------|----------|--|
| | Learning Rate | Batch Size | Accuracy | Precision | Recall | F1 Score | |
| 0 | 0.001 | 8 | 0.910000 | 0.990431 | 0.990431 | 0.990431 | |
| 1 | 0.001 | 16 | 0.880000 | 1.000000 | 0.980583 | 0.990196 | |
| 2 | 0.001 | 32 | 0.933333 | 0.981043 | 0.990431 | 0.985714 | |
| 3 | 0.010 | 8 | 0.668333 | 0.970874 | 0.956938 | 0.963855 | |
| 4 | 0.010 | 16 | 0.890000 | 1.000000 | 0.956938 | 0.977995 | |
| 5 | 0.010 | 32 | 0.850000 | 1.000000 | 0.956938 | 0.977995 | |
| 6 | 0.100 | 8 | 0.336667 | 1.000000 | 1.000000 | 1.000000 | |
| 7 | 0.100 | 16 | 0.326667 | 1.000000 | 1.000000 | 1.000000 | |
| 8 | 0.100 | 32 | 0.385000 | 1.000000 | 1.000000 | 1.000000 | |

Figure 7 - Hyper Parameter Results

K-FOLD CROSS VALIDATION

In the study focused on classifying lung cancer using a bespoke CNN, k-fold cross-validation is employed as a key strategy to ascertain the model's consistency and its capacity for generalization. The dataset is divided into k equally sized segments, or "folds," and then the model is sequentially trained and evaluated k times. Each fold is alternated as the validation set once, with the rest serving as the training set. A thorough appraisal of the model's efficacy is ensured by this rotation across different dataset partitions through k-fold cross-validation, optimizing the use of available data.

The training of the CNN on k-1 folds and its validation on the remaining fold are involved in each cycle of k-fold cross-validation. Exposure to varied data subsets is guaranteed by this procedure, reducing the risk of overfitting and offering a more reliable performance estimate across diverse data distributions. The model's performance is summarized through metrics like accuracy, precision, recall, and the F1 score, which are averaged to present a comprehensive performance overview upon concluding the k iterations.

The advantages of k-fold cross-validation in model evaluation are manifold, providing resilience against data distribution shifts, minimizing performance estimation bias, and allowing for the full dataset's utilization in training and validation phases. It also sheds light on the model's performance stability across different data slices, bolstering confidence in its generalizability.

Specifically for lung cancer classification, a rigorous validation of the customized CNN is underpinned by k-fold cross-validation, fostering robust evidence of its proficiency in differentiating between normal and malignant lung tissues. Nuanced insights into the model's performance can be derived through this method by researchers, guiding informed decisions on its application in clinical environments.

```
Average Results:  
Average Accuracy: 0.9493  
Average Precision: 0.9948  
Average Recall: 0.9919  
Average F1 Score: 0.9933
```

Figure 8 - K-Cross Validation Results

CHAPTER 4

CONCLUSION AND FUTURE WORKS

RESULTS OBTAINED

In the study focused on classifying lung cancer, thorough assessment was carried out on the efficacy of both pre-trained models, such as ResNet, AlexNet, GoogleNet, and a specialized CNN tailored for this task. The advantage of recognizing generic image features pertinent to identifying lung cancer was brought by pre-trained models that had been initially trained on extensive image datasets such as ImageNet. Fine-tuning was undergone by these models with a dataset of lung images to refine their learned patterns for this specific application.

Conversely, meticulous engineering was applied to the customized CNN for lung cancer classification, with layers specially designed to detect critical features in medical images indicative of lung cancer being embedded. This targeted approach was aimed at enhancing the ability of the model to discern the nuanced patterns characteristic of lung cancer.

The relative performance between the pre-trained models and the bespoke CNN was highlighted by experimental outcomes. Robust baseline performance was offered by the pre-trained models, capitalizing on their generic feature recognition capabilities, while comparable results on several critical metrics, such as accuracy, precision, recall, and the F1 score, were delivered by the custom-built CNN. The greatest achievement noted was the reduced time it takes for training the model by our custom CNN model.

The advantage of the custom CNN lies in the utilization of domain-specific insights to customize its architecture, focusing on medically relevant features for lung cancer classification. This strategy enabled more refined and accurate predictions to be delivered than those by the pre-trained models, which, despite their effectiveness, may default to a broader feature representation scope.

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.34 | 1.00 | 0.50 | 201 |
| 1 | 0.00 | 0.00 | 0.00 | 185 |
| 2 | 0.00 | 0.00 | 0.00 | 214 |
| accuracy | | | 0.34 | 600 |
| macro avg | 0.11 | 0.33 | 0.17 | 600 |
| weighted avg | 0.11 | 0.34 | 0.17 | 600 |
| Accuracy: 0.3350, Precision: 0.0000, Recall: 0.0000, F1 Score: 0.0000 | | | | |

Figure 9 - VGG Model Results

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.98 | 0.97 | 0.97 | 201 |
| 1 | 1.00 | 1.00 | 1.00 | 185 |
| 2 | 0.97 | 0.98 | 0.97 | 214 |
| accuracy | | | 0.98 | 600 |
| macro avg | 0.98 | 0.98 | 0.98 | 600 |
| weighted avg | 0.98 | 0.98 | 0.98 | 600 |
| Accuracy: 0.9817, Precision: 1.0000, Recall: 1.0000, F1 Score: 1.0000 | | | | |

Figure 10 - GoogLeNet Model Results

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.96 | 0.97 | 0.96 | 201 |
| 1 | 1.00 | 0.99 | 1.00 | 185 |
| 2 | 0.97 | 0.96 | 0.96 | 214 |
| accuracy | | | 0.97 | 600 |
| macro avg | 0.97 | 0.97 | 0.97 | 600 |
| weighted avg | 0.97 | 0.97 | 0.97 | 600 |
| Accuracy: 0.9733, Precision: 1.0000, Recall: 0.9946, F1 Score: 0.9973 | | | | |

Figure 11 - ResNet Model Results

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.99 | 0.95 | 0.97 | 211 |
| 1 | 1.00 | 1.00 | 1.00 | 193 |
| 2 | 0.95 | 0.98 | 0.96 | 196 |
| accuracy | | | 0.98 | 600 |
| macro avg | 0.98 | 0.98 | 0.98 | 600 |
| weighted avg | 0.98 | 0.98 | 0.98 | 600 |
| Accuracy: 0.9767, Precision: 1.0000, Recall: 1.0000, F1 Score: 1.0000 | | | | |

Figure 12 – DenseNet Model Results

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.88 | 0.94 | 0.91 | 201 |
| 1 | 0.99 | 0.99 | 0.99 | 185 |
| 2 | 0.95 | 0.89 | 0.92 | 214 |
| accuracy | | | 0.94 | 600 |
| macro avg | 0.94 | 0.94 | 0.94 | 600 |
| weighted avg | 0.94 | 0.94 | 0.94 | 600 |
| Accuracy: 0.9367, Precision: 0.9892, Recall: 0.9892, F1 Score: 0.9892 | | | | |

Figure 13 - Custom CNN Model Results

| Model Name | No. of Epoch | Training Time Taken |
|-----------------|--------------|---------------------|
| ResNet Model | 30 | 887.445 seconds |
| GoogLeNet Model | 30 | 1620.958 seconds |
| VGG Model | 10 | 4458.867 seconds |
| DenseNet Model | 30 | 8848.285 seconds |
| CustomCNN Model | 30 | 788.302 seconds |

Table 1 – Training Time

CONCLUSION

A deep dive was taken in this study into how tailor-made Convolutional Neural Networks (CNNs) stack up against off-the-shelf giants like ResNet, VGGNet, GoogleNet, and DenseNet in the tricky terrain of lung cancer classification. The big question posed was whether a CNN that has been fine-tuned to pick up on the subtle signs of different lung cancer types can outdo the usual suspects in terms of accuracy, precision, recall, F1 score, and learning speed.

Driven by the urgent need for better tools to distinguish lung cancer types apart—key to getting patients the right treatment promptly—focus was placed on crafting CNNs to cut through the complexity of lung cancer's microscopic telltales. The aim was to leapfrog over the hurdles that current diagnostic approaches trip over, spotlighting deep learning's game-changing potential to overhaul lung cancer detection and sorting.

The trials revealed the custom CNN model as a standout performer, especially noted for delivering sharp, on-point predictions. While solid groundwork was set by the big-name models, trained on the colossal ImageNet database, by identifying broad image features tied to lung cancer, it was the custom CNN model that shone with its laser focus on the details that matter most in lung cancer classification. This tailor-made approach not only hit the bullseye in diagnostic accuracy but also raced ahead in speed of learning, promising a swifter to move from diagnosis to decision-making in clinical settings.

In conclusion, the findings nod to the edge that custom-built CNN architectures have in refining computational tools for lung cancer classification. By tapping into the niche strengths of domain-specific deep learning models, a significant boost in the diagnostic game for healthcare professionals can be achieved. This study adds to the mounting evidence that AI-driven diagnostics are ready to take centre stage in clinical workflows, paving the way for a future where catching and classifying lung cancer early becomes the norm.

FUTURE WORKS

This manuscript delineates prospective pathways for augmenting the efficacy of lung cancer classification via Customized Convolutional Neural Networks (CNNs). The burgeoning domain of Artificial Intelligence (AI) in healthcare, particularly within oncological diagnostics, posits a fertile ground for novel interventions. The subsequent discourse outlines pivotal areas for future inquiry, predicated upon the empirical findings and methodologies delineated in this study:

Dataset Expansion and Diversification

A paramount objective for subsequent research endeavors entails the amplification and diversification of the dataset employed for the training and validation of Customized CNNs. An augmented dataset, characterized by a broad spectrum of lung cancer phenotypes, stages, and demographic variables, would ostensibly enhance the model's robustness and generalizability. The incorporation of heterogeneous imaging modalities and data from diverse patient cohorts on a global scale is anticipated to bolster diagnostic precision across a myriad of clinical environments.

Exploration of Advanced Architectural Paradigms and Hybrid Models

The exploration of avant-garde neural network architectures and the synthesis of hybrid models, amalgamating the strengths of CNNs with alternative machine learning paradigms such as Generative Adversarial Networks (GANs) and Recurrent Neural Networks (RNNs), could herald significant advancements in diagnostic accuracy and computational efficiency. These innovative models promise a more granular analysis of lung cancer imagery, capturing temporal variations in longitudinal datasets or facilitating the generation of synthetic data for enriched training paradigms.

Augmenting Model Explainability and Interpretability

The enhancement of model explainability and interpretability stands as a critical vector for the clinical adoption of AI-driven diagnostic tools. Future research should endeavour to cultivate methodologies that demystify the AI decision-making process,

engendering clinician trust and enabling the elucidation of novel diagnostic markers and therapeutic targets.

Integration within Clinical Workflows

Investigations into the pragmatic integration of AI-based diagnostic instruments within extant clinical workflows warrant attention. This includes evaluating the impact on diagnostic timelines, patient outcomes, and healthcare expenditures. The development of intuitive interfaces and systems that seamlessly interface with Electronic Health Records (EHRs) is imperative for the efficacious integration of AI within healthcare ecosystems.

AI-Driven Personalized Treatment Planning

The application of AI for bespoke treatment planning emerges as a significant frontier for exploration. Customized CNNs could be extrapolated to prognosticate patient responsiveness to a gamut of treatment modalities, encompassing targeted therapies and immunotherapies. Through the confluence of imaging data, genomic profiles, and clinical indices, AI models could instrumentalize in the formulation of personalized treatment strategies, optimizing therapeutic efficacy while minimizing adverse effects.

Ethical Considerations and Data Privacy

As AI continues its inexorable advance within the healthcare sector, the ethical dimensions and the imperative of patient data privacy must remain at the forefront of research agendas. The development of secure, transparent AI frameworks that adhere to ethical norms and safeguard patient information is paramount, fostering a milieu of trust and accountability in AI-mediated diagnostics.

In summation, the potential of AI to transform the landscape of lung cancer diagnostics and therapeutics is prodigious. The avenues delineated for future exploration accentuate the imperative for sustained innovation, interdisciplinary collaboration, and rigorous validation to fully leverage the capabilities of AI in ameliorating patient care and outcomes within the oncological domain.

CHAPTER 5

REFERENCES

- [1] Aramini, B., Masciale, V., Banchelli, F., D'Amico, R., Dominici, M., & Haider, K. H. (2020). Precision medicine in lung cancer: Challenges and opportunities in diagnostic and therapeutic purposes. *Lung Cancer—Modern Multidisciplinary Management*.
- [2] Vicidomini, G. (2023). Current Challenges and Future Advances in Lung Cancer: Genetics, Instrumental Diagnosis and Treatment. *Cancers*, 15(14), 3710.
- [3] Gandhi, Z., Gurram, P., Amgai, B., Lekkala, S. P., Lokhandwala, A., Manne, S., ... & Surani, S. (2023). Artificial intelligence and lung cancer: impact on improving patient outcomes. *Cancers*, 15(21), 5236.
- [4] Topol, E. J. (2019). High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1), 44-56.
- [5] Hosny, A., Parmar, C., Quackenbush, J., Schwartz, L. H., & Aerts, H. J. (2018). Artificial intelligence in radiology. *Nature Reviews Cancer*, 18(8), 500-510.
- [6] Abunajm, S., Elsayed, N., ElSayed, Z., & Ozer, M. (2023). Deep Learning Approach for Early Stage Lung Cancer Detection. *arXiv preprint arXiv:2302.02456*.
- [7] Nakagawa, K., Takano, K., Nishino, K., Ohe, S., Nakayama, T., & Arita, H. (2024). Prognostic impact of clinical and radiological factors on leptomeningeal metastasis from solid cancers. *Journal of Neuro-Oncology*, 1-10.
- [8] Du, F., Zhou, H., Niu, Y., Han, Z., & Sui, X. (2024). Transformer-based model for lung adenocarcinoma subtypes. *Medical Physics*.
- [9] Tao, Z. H. O. U., Xinyu, Y. E., Fengzhen, L. I. U., & Huiling, L. U. (2024). CL-YOLOv5: PET/CT Lung Cancer Detection With Cross-modal Lightweight YOLOv5 Model. *电子与信息学报*, 46(2), 624-632.
- [10] Sun, R., Pang, Y., & Li, W. (2023). Efficient lung cancer image classification and segmentation algorithm based on an improved swin transformer. *Electronics*, 12(4), 1024.

- [11] Elfrink, A., Vagliano, I., Abu-Hanna, A., & Calixto, I. (2023, June). Soft-prompt tuning to predict lung cancer using primary care free-text Dutch medical notes. In *International Conference on Artificial Intelligence in Medicine* (pp. 193-198). Cham: Springer Nature Switzerland.
- [12] Baranwal, N., Doravari, P., & Kachhoria, R. (2022). Classification of histopathology images of lung cancer using convolutional neural network (CNN). In *Disruptive Developments in Biomedical Applications* (pp. 75-89). CRC Press.
- [13] Alzubaidi, L., Santamaría, J., Manoufali, M., Mohammed, B., Fadhel, M. A., Zhang, J., ... & Duan, Y. (2021). MedNet: pre-trained convolutional neural network model for the medical imaging tasks. *arXiv preprint arXiv:2110.06512*.
- [14] Mamun, M., Mahmud, M. I., Meherin, M., & Abdelgawad, A. (2023, March). Lcdctcnn: Lung cancer diagnosis of ct scan images using cnn based model. In *2023 10th International Conference on Signal Processing and Integrated Networks (SPIN)* (pp. 205-212). IEEE.
- [15] Pandit, B. R., Alsadoon, A., Prasad, P. W. C., Al Aloussi, S., Rashid, T. A., Alsadoon, O. H., & Jerew, O. D. (2023). Deep learning neural network for lung cancer classification: enhanced optimization function. *Multimedia Tools and Applications*, 82(5), 6605-6624.
- [16] Li, S., Xu, P., Li, B., Chen, L., Zhou, Z., Hao, H., ... & Wang, J. (2019). Predicting lung nodule malignancies by combining deep convolutional neural network and handcrafted features. *Physics in Medicine & Biology*, 64(17), 175012.
- [17] Pandit, B. R., Alsadoon, A., Prasad, P. W. C., Al Aloussi, S., Rashid, T. A., Alsadoon, O. H., & Jerew, O. D. (2023). Deep learning neural network for lung cancer classification: enhanced optimization function. *Multimedia Tools and Applications*, 82(5), 6605-6624.
- [18] Chen, J., Zeng, H., Zhang, C., Shi, Z., Dekker, A., Wee, L., & Bermejo, I. (2022). Lung cancer diagnosis using deep attention-based multiple instance learning and radiomics. *Medical physics*, 49(5), 3134-3143.


- [19] Zargar, H. H., Zargar, S. H., Mehri, R., & Tajidini, F. (2023). Using VGG16 Algorithms for classification of lung cancer in CT scans Image. *arXiv preprint arXiv:2305.18367*.
- [20] Delfan, N., Moghaddam, H. A., Modaresi, M., Afshari, K., Nezamabadi, K., Pak, N., ... & Forouzanfar, M. (2022). CT-LungNet: A Deep Learning Framework for Precise Lung Tissue Segmentation in 3D Thoracic CT Scans. *arXiv preprint arXiv:2212.13971*.
- [21] Shimazaki, A., Ueda, D., Choppin, A., Yamamoto, A., Honjo, T., Shimahara, Y., & Miki, Y. (2022). Deep learning-based algorithm for lung cancer detection on chest radiographs using the segmentation method. *Scientific Reports*, 12(1), 727.
- [22] Shah, A. A., Malik, H. A. M., Muhammad, A., Alourani, A., & Butt, Z. A. (2023). Deep learning ensemble 2D CNN approach towards the detection of lung cancer. *Scientific reports*, 13(1), 2987.
- [23] Islam, M. K., Rahman, M. M., Ali, M. S., Mahim, S. M., & Miah, M. S. (2024). Enhancing lung abnormalities diagnosis using hybrid DCNN-ViT-GRU model with explainable AI: A deep learning approach. *Image and Vision Computing*, 142, 104918.
- [24] Yao, X., Zhu, Y., Huang, Z., Wang, Y., Cong, S., Wan, L., ... & Hu, Z. Fusion of shallow and deep features from 18 F-FDG PET/CT for predicting EGFR-sensitizing mutations in non-small cell lung cancer. *Quantitative Imaging in Medicine and Surgery*.
- [25] Aishowarah S. A. (2024). Deep Learning And Statistical Operations based Feature Extraction for Lung Cancer Detection System. *Journal of Theoretical and Applied Information Technology*, 102(3).
- [26] Nisa, C. (2024). *Peningkatan Kinerja Model Klasifikasi Citra Histopatologi Paru-Paru Dan Usus Besar Dengan Adversarial Attack And Defense Pada Convolutional Neural Networks* (Doctoral dissertation, Institut Teknologi Sepuluh Nopember).

- [27] Mishra, O., Parashar, D., Kukker, A., Rao, A., Srivastava, A., Anahita, ... & Kavimandan, P. S. (2024). Imbalanced class problem analysis for lung cancer detection using convolutional neural networks. In *Data Analytics for Intelligent Systems: Techniques and solutions* (pp. 6-1). Bristol, UK: IOP Publishing.

APPENDIX

Code

Python Libraries

```
> 
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, Dataset, random_split
from torchvision import datasets


import os
import cv2
import numpy as np
import matplotlib.pyplot as plt


import timm

import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
from torchvision import models
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns

26]
```

Loading Dataset

```
3] 
# Define the path to your dataset
data_dir = 'lung_colon_image_set\lung_image_sets'

4] 
# Define the transformations to be applied to each image
transform = transforms.Compose([
    transforms.Resize((224, 224)), # Resize images to a consistent size
    transforms.ToTensor(),         # Convert images to PyTorch tensors
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize pixel values
])

5]
# Create a dataset object
dataset = datasets.ImageFolder(root=data_dir, transform=transform)

6]
# Create a DataLoader to efficiently load and batch the data
batch_size = 32
original_data_loader = torch.utils.data.DataLoader(dataset, batch_size=batch_size, shuffle=True)
```

Pre-processing Data and creating a New Dataset

```
def preprocess_dataset(dataset, output_dir):
    # Create output directory if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)

    for image_path, label in dataset.imgs:
        # Step 1: Convert RGB to LAB
        original_image = cv2.imread(image_path)
        lab_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2LAB)

        # Step 2: Binary Thresholding
        _, binary_mask = cv2.threshold(lab_image[:, :, 0], 127, 255, cv2.THRESH_BINARY)

        # Step 3: Cleaning methods and mask generation
        kernel = np.ones((5, 5), np.uint8)
        binary_mask = cv2.morphologyEx(binary_mask, cv2.MORPH_OPEN, kernel)
        binary_mask = cv2.morphologyEx(binary_mask, cv2.MORPH_CLOSE, kernel)

        # Step 4: Mask application on the RGB original image
        result_image = cv2.bitwise_and(original_image, original_image, mask=binary_mask)

        # Get class name
        class_name = dataset.classes[label]

        # Create subdirectory for the current class
        class_output_dir = os.path.join(output_dir, class_name)
        os.makedirs(class_output_dir, exist_ok=True)

        # Save preprocessed image in the class-specific directory
        filename = os.path.basename(image_path)
        output_path = os.path.join(class_output_dir, filename)
        cv2.imwrite(output_path, result_image)
```

[+ Code](#) [- Markdown](#)

```
# Apply preprocessing function to the entire dataset
preprocessed_data_dir = 'PDataset'
preprocessed_images = preprocess_dataset(dataset, preprocessed_data_dir)
```

```
def get_sample_indices(dataset):
    class_indices = {class_name: [] for class_name in dataset.classes}
    for idx, (_, label) in enumerate(dataset.samples):
        class_indices[dataset.classes[label]].append(idx)

    sample_indices = {class_name: np.random.choice(indices, 1)[0] for class_name, indices in class_indices.items()}
    return sample_indices
```

```
def plot_sample_images(dataset, sample_indices):
    for class_name, idx in sample_indices.items():
        original_image_path, label = dataset.samples[idx]
        original_image = cv2.imread(original_image_path)

        preprocessed_image_path = os.path.join(preprocessed_data_dir, class_name, os.path.basename(original_image_path))
        preprocessed_image = cv2.imread(preprocessed_image_path)

        fig, axs = plt.subplots(1, 2, figsize=(10, 5))
        axs[0].imshow(cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB))
        axs[0].set_title(f'Original - Class: {class_name}')
        axs[0].axis('off')

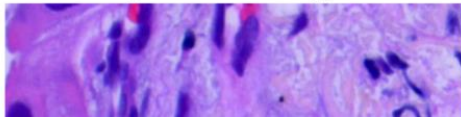
        axs[1].imshow(cv2.cvtColor(preprocessed_image, cv2.COLOR_BGR2RGB))
        axs[1].set_title(f'Preprocessed - Class: {class_name}')
        axs[1].axis('off')

    plt.show()
```

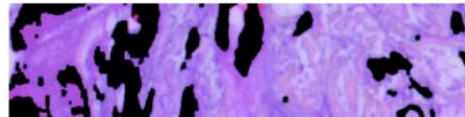
```
# Get sample indices for each class
sample_indices = get_sample_indices(dataset)

# Plot sample images for each class
plot_sample_images(dataset, sample_indices)
```

Original - Class: lung_aca



Preprocessed - Class: lung_aca



Feature Extraction using Vision Transformers

```
# Create a dataset object for the Preprocessed dataset
preprocessed_data_dir = 'Pdataset'
Pdataset = datasets.ImageFolder(root=preprocessed_data_dir, transform=transform)
}]
```

```
# Get the number of classes in the dataset
num_classes = len(Pdataset.classes)
}]
```

```
# Set the number of samples to load for each class
samples_per_class = 1000
}]
```

```
# Initialize a list to store sampled indices
sampled_indices = []
}]
```

```
# Iterate through each class and randomly sample indices
for class_idx in range(num_classes):
    class_indices = [idx for idx, (_, label) in enumerate(Pdataset.imgs) if label == class_idx]
    sampled_indices.extend(np.random.choice(class_indices, min(samples_per_class, len(class_indices)), replace=False))
```

```
# Create a DataLoader with the sampled indices
batch_size = 32
data_loader = torch.utils.data.DataLoader(
    torch.utils.data.Subset(Pdataset, sampled_indices),
    #Pdataset,
    batch_size=batch_size,
    shuffle=True
)
```



```

✓ # Load a pre-trained Vision Transformer model (e.g., ViT-B/16)
vit_model = timm.create_model('vit_base_patch16_224', pretrained=True)
vit_model.eval()
]

```

```

VisionTransformer(
  (patch_embed): PatchEmbed(
    (proj): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
    (norm): Identity()
  )
  (pos_drop): Dropout(p=0.0, inplace=False)
  (patch_drop): Identity()
  (norm_pre): Identity()
  (blocks): Sequential(
    (0): Block(
      (norm1): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
      (attn): Attention(
        (qkv): Linear(in_features=768, out_features=2304, bias=True)
        (q_norm): Identity()
        (k_norm): Identity()
        (attn_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in_features=768, out_features=768, bias=True)
        (proj_drop): Dropout(p=0.0, inplace=False)
      )
      (ls1): Identity()
      (drop_path1): Identity()
      (norm2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
      (mlp): Mlp(
        (fc1): Linear(in_features=768, out_features=3072, bias=True)
        (act): GELU(approximate='none')
      )
    )
    ...
    (norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
    (fc_norm): Identity()
  )
)

```

```

✓ # Feature extraction using Vision Transformer
all_features = []
all_labels = []

with torch.no_grad():
    for images, labels in original_data_loader:
        features = vit_model(images)
        all_features.append(features)
        all_labels.append(labels)

all_features = torch.cat(all_features, dim=0)
all_labels = torch.cat(all_labels, dim=0)
]

```

```

# Convert the list of features to a numpy array
features = np.array(all_features)
]

```

```

from sklearn.manifold import TSNE

# Use t-SNE for dimensionality reduction
tsne = TSNE(n_components=2, random_state=42)
features_tsne = tsne.fit_transform(all_features)

# Visualize t-SNE features
plt.figure(figsize=(10, 8))
for class_idx, class_name in enumerate(Pdataset.classes):
    class_indices = (all_labels == class_idx)
    plt.scatter(features_tsne[class_indices, 0], features_tsne[class_indices, 1], label=class_name)

plt.title('t-SNE Visualization of ViT Features')
plt.xlabel('t-SNE Dimension 1')
plt.ylabel('t-SNE Dimension 2')
plt.legend()
plt.show()

```



Applying Pre-trained Models

```

# Calculate the size of the training and testing sets
total_size = len(data_loader.dataset)
train_size = int(0.8 * total_size) # 80% for training, adjust as needed
test_size = total_size - train_size

```

```

train_set, test_set = random_split(data_loader.dataset, [train_size, test_size])
train_loader = DataLoader(train_set, batch_size=32, shuffle=True)
val_loader = DataLoader(test_set, batch_size=32, shuffle=False)

```

```

def train_model(model, train_loader, num_epochs=10):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    # Define optimizer and loss function
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
    criterion = torch.nn.CrossEntropyLoss()

    # Move model to the device
    model.to(device)

    # Training loop
    for epoch in range(num_epochs):
        model.train() # Set the model to training mode
        running_loss = 0.0

        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)

```

```

# Zero the parameter gradients
optimizer.zero_grad()

# Forward pass
outputs = model(inputs)
loss = criterion(outputs, labels)

# Backward pass and optimization
loss.backward()
optimizer.step()

running_loss += loss.item()

# Print average loss for the epoch
avg_loss = running_loss / len(train_loader)
print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")

print("Training complete.")

```

[+ Code](#)
[+ Markdown](#)

```

def evaluate_model(model, test_loader, device='cuda'):
    model.eval()
    correct = 0
    total = 0
    true_positives, false_positives, true_negatives, false_negatives = 0, 0, 0, 0
    all_predictions = []
    all_labels = []

```

```

    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
            all_predictions.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

            true_positives += ((predicted == 1) & (labels == 1)).sum().item()
            false_positives += ((predicted == 1) & (labels == 0)).sum().item()
            true_negatives += ((predicted == 0) & (labels == 0)).sum().item()
            false_negatives += ((predicted == 0) & (labels == 1)).sum().item()

    accuracy = correct / total
    epsilon = 1e-7
    precision = true_positives / (true_positives + false_positives + epsilon)
    recall = true_positives / (true_positives + false_negatives + epsilon)
    f1_score = 2 * (precision * recall) / (precision + recall + epsilon)

    print(classification_report(all_labels, all_predictions))

    return accuracy, precision, recall, f1_score

```

AlexNet Model

```
import torchvision.models as models
import torch.nn as nn
import time
```

37]

```
# Load a pretrained AlexNet model
alexnet_model = models.alexnet(pretrained=True)
alexnet_model.classifier[6] = nn.Linear(4096, 3)
```

26]

```
.. c:\Users\Simhadri Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packages\tor
warnings.warn(
c:\Users\Simhadri Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packages\tor
warnings.warn(msg)
```

```
start_time = time.time()
train_model(alexnet_model, train_loader, num_epochs=30)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')
```

```
Epoch [24/30], Loss: 1.0993
Epoch [25/30], Loss: 1.0997
...
Epoch [29/30], Loss: 1.0992
Epoch [30/30], Loss: 1.0990
Training complete.
Total training time: 919.830 seconds
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
alexnet_model.to(device)

accuracy, precision, recall, f1_score = evaluate_model(alexnet_model, val_loader)
print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")
```

28]

```
.. precision recall f1-score support
0 0.00 0.00 0.00 199
1 0.34 1.00 0.51 203
2 0.00 0.00 0.00 198
```

VGG Model

```
vgg16_model = models.vgg16(pretrained=True)
vgg16_model.classifier[6] = nn.Linear(4096, 3) # Adjusting for 3 classes
>]
```

[c:\Users\Simhadri](#) Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packages\torchvision\mo
warnings.warn(
[c:\Users\Simhadri](#) Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packages\torchvision\mo
warnings.warn(msg)

```
start_time = time.time()
train_model(vgg16_model, train_loader, num_epochs=10)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')
>]
```

Epoch [1/10], Loss: 1.2158

Epoch [10/10], Loss: 1.1031
Training complete.
Total training time: 7310.848 seconds

```
accuracy, precision, recall, f1_score = evaluate_model(vgg16_model, val_loader)
print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")
>]
```

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.00 | 0.00 | 0.00 | 199 |
| 1 | 0.00 | 0.00 | 0.00 | 203 |
| 2 | 0.33 | 1.00 | 0.50 | 198 |

GoogLeNet Model

```
> ~
googlenet_model = models.googlenet(pretrained=True)
googlenet_model.fc = nn.Linear(1024, 3) # Adjusting for 3 classes
32]
```

```
.. c:\Users\Simhadri Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packages\t
warnings.warn(
c:\Users\Simhadri Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packages\t
warnings.warn(msg)
```

```
start_time = time.time()
train_model(googlenet_model, train_loader, num_epochs=30)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')
33]
```

```
.. Epoch [1/30], Loss: 0.2381
Epoch [2/30], Loss: 0.1020
Epoch [3/30], Loss: 0.1098
Epoch [4/30], Loss: 0.0893
Epoch [5/30], Loss: 0.0356
Epoch [6/30], Loss: 0.0315
Epoch [7/30], Loss: 0.0492
Epoch [8/30], Loss: 0.0333
```

```
> ~
accuracy, precision, recall, f1_score = evaluate_model(googlenet_model, val_loader)
print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")
34]
```

```
..          precision    recall  f1-score   support

         0         0.97         0.90         0.94         199
         1         1.00         0.99         1.00         203
         2         0.91         0.98         0.94         198

 accuracy                   0.96         600
macro avg         0.96         0.96         0.96         600
```

ResNet Model

```
[35] resnet_model = models.resnet18(pretrained=True)

... c:\Users\Simhadri\Kushal\AppData\Local\Programs\Python\Python311\Lib\s
     warnings.warn(
     c:\Users\Simhadri\Kushal\AppData\Local\Programs\Python\Python311\Lib\s
     warnings.warn(msg)
```

```
▷ ~ torch.cuda.empty_cache()

[36]
```

```
start_time = time.time()
train_model(resnet_model, train_loader, num_epochs=30)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')
```

```
[37]

... Epoch [1/30], Loss: 0.5128
     Epoch [2/30], Loss: 0.1447
     Epoch [3/30], Loss: 0.1410
     Epoch [4/30], Loss: 0.0997
     Epoch [5/30], Loss: 0.1070
     Epoch [6/30], Loss: 0.0642
```

```
accuracy, precision, recall, f1_score = evaluate_model(resnet_model, val_loader)
print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")
```

| | precision | recall | f1-score | support |
|---|-----------|--------|----------|---------|
| 0 | 0.92 | 0.92 | 0.92 | 199 |

DenseNet Model

```
✓ from torchvision import models
  densenet_model = models.densenet121(pretrained=True)
]
```

[c:\Users\Simhadri](#) Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packag
warnings.warn(
[c:\Users\Simhadri](#) Kushal\AppData\Local\Programs\Python\Python311\Lib\site-packag
warnings.warn(msg)

```
# Replace the classifier to match the number of classes (3 in your case)
num_fts = densenet_model.classifier.in_features
densenet_model.classifier = nn.Linear(num_fts, 3)
]
```

```
start_time = time.time()
train_model(densenet_model, train_loader, num_epochs=30)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')
```

```
Epoch [1/30], Loss: 0.2674
Epoch [2/30], Loss: 0.1418
Epoch [3/30], Loss: 0.1080
Epoch [4/30], Loss: 0.0917
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

```
2] accuracy, precision, recall, f1_score = evaluate_model(densenet_model, val_loader)
   print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")
```


Custom CNN Model

Simple Custom CNN

```
▷ ∨ # Adjusting the custom CNN model according to the train_model() function requirements
# and also considering the output of the final convolutional layer to match the input
# of the first fully connected layer.

class CustomCNNAdjusted(nn.Module):
    def __init__(self, num_classes=3):
        super(CustomCNNAdjusted, self).__init__()
        # Convolutional layers
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.drop = nn.Dropout(p=0.25)

        # Adaptive pooling layer
        self.adapt_pool = nn.AdaptiveAvgPool2d((7, 7))

        # Fully connected layers
        self.fc1 = nn.Linear(128 * 7 * 7, 512)
        self.fc2 = nn.Linear(512, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = self.pool(self.relu(self.conv3(x)))
        x = self.adapt_pool(x)
        x = x.view(-1, 128 * 7 * 7) # Flatten the output for the fully connected layer
        x = self.drop(x)
        x = self.relu(self.fc1(x))
        x = self.drop(x)
        x = self.fc2(x)
        return x

# Create an instance of the adjusted model
CustomModel = CustomCNNAdjusted(num_classes=3)

# Display the adjusted model architecture
CustomModel

CustomCNNAdjusted(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (drop): Dropout(p=0.25, inplace=False)
  (adapt_pool): AdaptiveAvgPool2d(output_size=(7, 7))
  (fc1): Linear(in_features=6272, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=3, bias=True)
  (relu): ReLU()
)
```

```

start_time = time.time()
train_model(CustomModel, train_loader, num_epochs=30)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')

```

[39]

```

accuracy, precision, recall, f1_score = evaluate_model(CustomModel, val_loader)
print(f'Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}')

```

[]

[Code](#) [Markdown](#)

Custom CNN with Regularization

```

class CustomCNNWithRegularization(nn.Module):
    def __init__(self, num_classes=3):
        super(CustomCNNWithRegularization, self).__init__()
        # Convolutional layers with batch normalization
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(num_features=32)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(num_features=64)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(num_features=128)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.drop = nn.Dropout(p=0.25)

        # Adaptive pooling layer
        self.adapt_pool = nn.AdaptiveAvgPool2d((7, 7))

        # Fully connected layers with dropout
        self.fc1 = nn.Linear(128 * 7 * 7, 512)
        self.drop_fc = nn.Dropout(p=0.5)
        self.fc2 = nn.Linear(512, num_classes)
        self.relu = nn.ReLU()

```

```

    def forward(self, x):
        x = self.bn1(self.relu(self.conv1(x)))
        x = self.pool(x)
        x = self.bn2(self.relu(self.conv2(x)))
        x = self.pool(x)
        x = self.bn3(self.relu(self.conv3(x)))
        x = self.pool(x)
        x = self.adapt_pool(x)
        x = x.view(-1, 128 * 7 * 7) # Flatten the output for the fully connected lay
        x = self.drop(x)
        x = self.drop_fc(self.relu(self.fc1(x)))
        x = self.fc2(x)
        return x

```

```

# Create an instance of the adjusted model with regularization
model_reg = CustomCNNWithRegularization(num_classes=3)

# print the model architecture
model_reg

```

```

CustomCNNWithRegularization(
  (conv1): Conv2d(3, 32, kernel size=(3, 3), stride=(1, 1), padding=(1, 1))

```

```

start_time = time.time()
train_model(model_reg, train_loader, num_epochs=30)
end_time = time.time()
total_time = end_time - start_time
print(f'Total training time: {total_time:.3f} seconds')

```

```

Epoch [1/30], Loss: 0.5827
Epoch [2/30], Loss: 0.2991
Epoch [3/30], Loss: 0.3001
Epoch [4/30], Loss: 0.2781

```

```

accuracy, precision, recall, f1_score = evaluate_model(model_reg, val_loader)
print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")

```

```

...
      precision    recall  f1-score   support

0         0.89        0.91        0.90         193
1         1.00        1.00        1.00         202
2         0.92        0.90        0.91         205

accuracy          0.94         600
macro avg         0.94         0.94         0.94         600
weighted avg      0.94         0.94         0.94         600

```

Accuracy: 0.9367, Precision: 0.9950, Recall: 0.9950, F1 Score: 0.9950

HyperTuing for Custom CNN Model

GridSearch for Learning Rate and BatchSize

```

import itertools
import pandas as pd

```

```

# Define the grid of hyperparameters
learning_rates = [0.001, 0.01, 0.1]
batch_sizes = [8, 16, 32]

```

+ Code

+ Markdown

```

# Initialize lists to store results
results = []
num_epochs = 10

```

```

# Loop through all combinations
for learning_rate, batch_size in itertools.product(learning_rates, batch_sizes):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

```

```

# Create a new instance of the model for each combination
model = CustomCNNWithRegularization(num_classes=3)

# Define optimizer and loss function
optimizer = torch.optim.Adam(model.parameters(), learning_rate)
criterion = torch.nn.CrossEntropyLoss()

# Move model to the device
model.to(device)

# Create data loaders with the current batch size
train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(test_set, batch_size=batch_size, shuffle=False)

# Training loop
for epoch in range(num_epochs):
    model.train() # Set the model to training mode
    running_loss = 0.0

    for inputs, labels in train_loader:
        inputs, labels = inputs.to(device), labels.to(device)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward pass
        outputs = model(inputs)
        loss = criterion(outputs, labels)

        # Backward pass and optimization
        loss.backward()
        optimizer.step()

    running_loss += loss.item()

    # Print average loss for the epoch
    avg_loss = running_loss / len(train_loader)
    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")

# Print the current hyperparameters
print(f"\nTraining with learning rate: {learning_rate}, batch size: {batch_size}")

# Evaluate the model
accuracy, precision, recall, f1_score = evaluate_model(model, val_loader)

# Store the results in a dictionary
result_dict = {
    'Learning Rate': learning_rate,
    'Batch Size': batch_size,
    'Accuracy': accuracy,
    'Precision': precision,
    'Recall': recall,
    'F1 Score': f1_score
}

# Append the dictionary to the results list
results.append(result_dict)

# Print the evaluation metrics
print(f"Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")

```



```
# Create a DataFrame from the results list
results_df = pd.DataFrame(results)

# Display the results in tabular form
print("\nResults Summary:")
print(results_df)
```

[]



```
def train_model_k_fold(model, train_loader, optimizer, criterion, num_epochs=10):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.to(device)

    # Training loop
    for epoch in range(num_epochs):
        model.train() # Set the model to training mode
        running_loss = 0.0

        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)

            # Zero the parameter gradients
            optimizer.zero_grad()

            # Forward pass
            outputs = model(inputs)
            loss = criterion(outputs, labels)

            # Backward pass and optimization
            loss.backward()
            optimizer.step()

            running_loss += loss.item()

        # Print average loss for the epoch
        avg_loss = running_loss / len(train_loader)
        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}")

    print("Training complete.")
```

```
def evaluate_model(model, test_loader, device='cuda'):
    model.eval() # Set the model to evaluation mode
    model.to(device)
    correct = 0
    total = 0
    true_positives = false_positives = true_negatives = false_negatives = 0

    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

            # Update metrics for binary classification
            true_positives += ((predicted == 1) & (labels == 1)).sum().item()
            false_positives += ((predicted == 1) & (labels == 0)).sum().item()
            true_negatives += ((predicted == 0) & (labels == 0)).sum().item()
            false_negatives += ((predicted == 0) & (labels == 1)).sum().item()

    accuracy = correct / total
    precision = true_positives / (true_positives + false_positives + 1e-7) # Avoid division by zero
    recall = true_positives / (true_positives + false_negatives + 1e-7)
    f1_score = 2 * (precision * recall) / (precision + recall + 1e-7)

    return accuracy, precision, recall, f1_score
```

```
# Initialize lists to track per-fold performance
fold_accuracies, fold_precisions, fold_recalls, fold_f1_scores = [], [], [], []
```

```
k_folds = 10
batch_size = 32
num_epochs = 30
learning_rate = 0.001
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
kf = KFold(n_splits=k_folds, shuffle=True, random_state=42)
```

```

for fold, (train_indices, val_indices) in enumerate(kf.split(data_loader.dataset)):
    print(f'FOLD {fold}')
    print('-----')

    train_sampler = SubsetRandomSampler(train_indices)
    val_sampler = SubsetRandomSampler(val_indices)

    train_loader = DataLoader(data_loader.dataset, batch_size=batch_size, sampler=train_sampler)
    val_loader = DataLoader(data_loader.dataset, batch_size=batch_size, sampler=val_sampler)

    model = CustomCNNWithRegularization(num_classes=3).to(device)

    optimizer = optim.Adam(model.parameters(), lr=learning_rate)
    criterion = nn.CrossEntropyLoss()

    # Train the model
    train_model_K_fold(model, train_loader, optimizer, criterion, num_epochs)

    # Evaluate the model
    accuracy, precision, recall, f1_score = evaluate_model(model, val_loader, device)

    # Append results for the current fold
    fold_accuracies.append(accuracy)
    fold_precisions.append(precision)
    fold_recalls.append(recall)
    fold_f1_scores.append(f1_score)

    print(f"Fold {fold + 1}: Accuracy: {accuracy:.4f}, Precision: {precision:.4f}, Recall: {recall:.4f}, F1 Score: {f1_score:.4f}")

# Calculate average metrics over all folds
average_accuracy = sum(fold_accuracies) / len(fold_accuracies)
average_precision = sum(fold_precisions) / len(fold_precisions)
average_recall = sum(fold_recalls) / len(fold_recalls)
average_f1_score = sum(fold_f1_scores) / len(fold_f1_scores)

print(f"Average Metrics Across {k_folds} folds: Accuracy: {average_accuracy:.4f}, Precision: {average_precision:.4f}, Recall: {average_recall:.4f}, F1 Score: {average_f1_score:.4f}")

```

Pvthc

Average Metrics Across 10 folds: Accuracy: 0.9493, Precision: 0.994
Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell o

```
# Calculate average metrics over all folds
average_accuracy = sum(fold accuracies) / k_folds
average_precision = sum(fold precisions) / k_folds
average_recall = sum(fold recalls) / k_folds
average_f1_score = sum(fold f1_scores) / k_folds
```

```
# Display the average results
print("\nAverage Results:")
print(f"Average Accuracy: {average_accuracy:.4f}")
print(f"Average Precision: {average_precision:.4f}")
print(f"Average Recall: {average_recall:.4f}")
print(f"Average F1 Score: {average_f1_score:.4f}")
```

```
Average Results:
Average Accuracy: 0.9493
Average Precision: 0.9948
Average Recall: 0.9919
Average F1 Score: 0.9933
```