POX DISEASE IDENTIFICATION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

ARAVIND KANNA V

(Reg.No:22BSR001)

GUGAN R S

(Reg.No:22BSR017)

NADISH G Y

(Reg.No:22BSR033)

In partial fulfillment of the requirement for the

Award of the degree

of

BACHELOR OF SCIENCE

IN

SOFTWARE SYSTEMS

DEPARTMENT OF COMPUTER TECHNOLOGY- UG

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

Estd : 1984

MARCH 2025

DEPARTMENT OF COMPUTER TECHNOLOGY – UG

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

MARCH 2025

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"POX DISEASE IDENTIFICATION USING DEEP LEARNING"** is the bonafide record of project work done by **ARAVIND KANNA V(Reg.No:22BSR001)** and **GUGAN R S (Reg. No: 22BSR017)** and **NADISH G Y (Reg. No: 22BSR033)** in partial fulfillment for the award of Degree of Bachelor of science in SOFTWARE SYSTEMS of Anna University, Chennai during the academic year 2024 -2025.

SUPERVISOR **HEAD OF THE DEPARTMENT**
(Signature with Seal)

Date:

Submitted for the end semester viva-voce examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We affirm that the project titled - **POX DISEASE IDENTIFICATION USING DEEP LEARNING‖** being submitted in partial fulfillment of the requirements for the award of Bachelor of Science in **SOFTWARE SYSTEMS** is the original work carried out by us. It has not formed part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this occasion on any other candidate.

ARAVIND KANNA V
(Reg.No:22BSR001)

GUGAN R S
(Reg.No:22BSR017)

NADISH G Y
(Reg.No:22BSR033)

DATE:

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor

ABSTRACT

Pox diseases, caused by viruses, pose significant health, economic, and agricultural challenges by affecting humans, animals, and plants. Early and accurate identification of these diseases is crucial for effective management and prevention. This study proposes a deep learning-based image classification approach to identify pox diseases by analyzing visual symptoms on affected subjects. By leveraging convolutional neural networks (CNNs), the model is trained on a diverse dataset of pox disease images, which accounts for variations in species, appearance, and environmental conditions.

The developed CNN model demonstrates high accuracy in distinguishing pox diseases from healthy conditions and other illnesses. Comprehensive performance evaluations highlight its robustness in terms of precision, recall, and computational efficiency. Compared to traditional machine learning techniques, the proposed method exhibits superior results, making it a reliable tool for automated disease diagnosis.

This research emphasizes the potential of deep learning to advance the identification and management of pox diseases. The findings provide a strong foundation for integrating AI- driven tools into disease detection workflows, ultimately enhancing early detection and decision-making capabilities. This work also paves the way for further exploration of deep learning applications in automated disease management systems.

ACKNOWLEDGEMENT

I respect and thank our Correspondent **Thiru.A.K.ILANGO BCom.,MBA., LLB.,** and our Principal **Dr.V.BALUSAMY BE (Hons).,MTech.,PhD.,** and all others trust members of Kongu Vellalar Institute of Technology Trust who have always encouraged us in the academic and co-curricular activities.

I convey my gratitude and heartfelt thanks to our Head of the Department **Dr.S.KALAISELVI MCA,ME,PhD.,** Department of Computer Technology-UG, Kongu Engineering College, for his perfect guidance and support that made this work to be completed successfully.

We are in immense pleasure to express our hearty thanks to our beloved Project coordinator **Dr.K. SARASWATHI Msc., MPhil., PhD.,** and our guide **Dr.S.KALAISELVI MCA, ME, PhD.,** for providing valuable guidance and constant support throughout the course of our project. We also thank the teaching, non-teaching staff members, fellow students and our parents who stood with us to complete our project successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
DL	Deep Learning
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
NASNet	Neural Architecture Search Network
DenseNet	Dense Convolutional Network

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Pox diseases remain a significant health concern worldwide, being one of the most common viral infections affecting both humans and animals. Despite advances in medical knowledge and treatment methods, the global incidence and prevalence of pox diseases continue to rise. This is primarily attributed to inadequate prevention, delayed detection, and suboptimal management strategies. The variability in symptoms and the lack of specific early warning signs make timely detection of pox diseases a challenge. Globally, outbreaks of pox diseases in both endemic and non-endemic areas cause considerable health risks and economic burdens, especially in agricultural and livestock sectors. Pox diseases are associated with fever, skin lesions, and in severe cases, organ failure or death, leading to a substantial burden on healthcare systems.

Deep learning (DL) models provide a transformative approach to predicting and managing pox diseases. By analysing patient data, including clinical symptoms and imaging, these models can identify high-risk individuals who might benefit from early intervention. Enhanced diagnostic tools leveraging DL algorithms can help healthcare providers detect pox diseases in their early stages. This proactive and personalized approach can lead to better treatment outcomes, reduced transmission rates, and improved patient quality of life

Beyond individual patient care, deep learning technologies also offer significant public health benefits. By analysing large datasets, DL models can identify trends and patterns in pox disease incidence across diverse populations. This information can guide policymakers in developing targeted public health strategies, such as promoting vaccination campaigns and improving surveillance systems to reduce the spread of pox diseases.

1.2 POX DISEASE

Pox diseases, such as chickenpox, cowpox and monkeypox, are viral infections characterized by the formation of skin lesions or pustules. These diseases are caused by viruses from the Poxviridae family and can affect both humans and animals. Pox diseases develop when the virus enters the body through respiratory droplets, direct contact with infected individuals, or contaminated surfaces. Once inside the body, the virus replicates, leading to symptoms such as fever, fatigue, and the appearance of distinctive pox lesions on the skin, which can vary in severity and distribution.

Pox diseases are a common infectious condition that can affect individuals of any age, though they are more prevalent in areas with limited healthcare access or vaccination coverage. Several factors contribute to the spread and severity of pox diseases, including close contact with infected individuals, compromised immune systems, and lack of vaccination. Environmental conditions, such as overcrowded living environments and poor sanitation, also increase the risk of transmission. Pox diseases can lead to severe complications if left untreated, including secondary bacterial infections, scarring, or, in severe cases, organ failure.

Preventive measures such as vaccination and maintaining proper hygiene are critical in controlling pox outbreaks. Early detection through health evaluations and rapid response to outbreaks are essential, particularly in populations at higher risk or during periods of increased viral activity. By understanding the causes, symptoms, and preventive strategies, individuals and communities can mitigate the spread of pox diseases and safeguard public health.

1.3 DEEP LEARNING

Deep learning is a transformative subset of machine learning and artificial intelligence (AI) that utilizes neural networks with multiple layers to model and analyse complex patterns in data. Inspired by the structure of the human brain, deep learning systems process data through interconnected layers of artificial neurons, automatically extracting and refining features to make predictions or decisions. Unlike traditional machine learning, deep learning requires minimal feature engineering, as it learns directly from raw data. This capability, coupled with its

ability to scale effectively with large datasets, has made deep learning indispensable in addressing modern data-driven challenges.

The applications of deep learning span a wide array of fields. In image and video analysis, it excels in tasks such as object detection, facial recognition, and medical imaging, revolutionizing areas like healthcare and security. Deep learning also plays a significant role in natural language processing (NLP), powering machine translation, chatbots, and virtual assistants like Siri and Alexa. Additionally, it is a cornerstone of autonomous systems like self-driving cars and drones, while also enabling advancements in speech recognition, finance, gaming, and personalized recommendations. Specialized architectures like Convolutional Neural Networks (CNNs) handle image data, Recurrent Neural Networks (RNNs) manage sequential data, and Transformers revolutionize text processing, further showcasing the versatility of deep learning.

Training deep learning models involves passing data through layers in a process known as forward propagation to generate predictions, followed by calculating the loss to measure errors. The model's weights are then updated using backpropagation, where gradients guide adjustments to improve accuracy. This iterative optimization process often relies on large datasets and high computational power, utilizing GPUs or TPUs for efficiency. Deep learning's ability to learn hierarchical representations and adapt to various data types has enabled groundbreaking advancements, from improving healthcare diagnostics to powering autonomous technologies. As research continues, the potential of deep learning to innovate and address societal challenges remains immense.

CHAPTER 2

LITERATURE SURVEY

The study "Computer-Aided Detection and Classification of Monkeypox and Chickenpox Lesion in Human Subjects Using Deep Learning Framework" by Dilber Uzun Ozsahin et al. explores the application of deep learning (DL) for accurately detecting and classifying monkeypox and chickenpox lesions from digital skin images, addressing the challenge of visual similarity between the lesions. By leveraging a custom Convolutional Neural Network (CNN) model with four convolutional layers and three MaxPooling layers, the study achieved a test accuracy of 99.60%, outperforming state-of-the-art models like AlexNet (98%) and VGGNet (80%). Image augmentation techniques were employed to enhance model generalization and avoid overfitting, demonstrating the effectiveness of DL in rapid and reliable diagnosis. The study highlights the potential of integrating DL models into clinical workflows to improve diagnostic accuracy and prevent misdiagnosis, paving the way for future enhancements, including optimizing the model for real-world deployment in healthcare [1].

The study "Deep and Transfer Learning Approaches for Automated Early Detection of Monkeypox (Mpox) Alongside Other Similar Skin Lesions and Their Classification" by Madhumita Pal et al. investigates the use of deep learning models, including CNN, VGG19, ResNet50, Inception v3, and Autoencoder, for the early and accurate detection of monkeypox (monkeypox) and differentiation from similar skin lesions like chickenpox and measles. Inception v3 outperformed other models with a classification accuracy of 96.56%, followed by VGG19 (94.06%) and CNN (93.43%), while Autoencoder showed the lowest accuracy at 85.62%. The study utilized a public dataset enhanced with image augmentation techniques to improve model robustness. By leveraging the superior feature extraction capability of Inception v3, the research underscores the feasibility of DL in real-time clinical diagnostics. It also explores future advancements, such as integrating IoT for automated diagnostics and expanding to video-based classification for enhanced accuracy [2].

The study "Monkeypox Virus Detection Using Pre-trained Deep Learning-based Approaches" by Chiranjibi Sitaula and Tej Bahadur Shahi compares 13 pre-trained deep learning models for monkeypox detection, fine-tuning them with custom layers to improve accuracy and performance. The research employs an ensemble learning approach, combining the best-performing models—Xception and DenseNet-169—using majority voting, achieving an average accuracy of 87.13%, which outperformed individual models. The dataset consisted of augmented images categorized into Monkeypox, Chickenpox, Measles, and Normal. The study demonstrates the effectiveness of ensemble DL models in accurate and early detection, supporting healthcare practitioners in mass screening. It also suggests future research directions, including enhancing model interpretability using Grad-CAM and LIME, and exploring advanced ensemble techniques for improved diagnostic performance [3].

The study "Detection of Monkeypox Among Different Pox Diseases with Different Pre-Trained Deep Learning Models" by Muhammed Çelik and Özkan İnik investigates the effectiveness of deep learning (DL) models in distinguishing monkeypox from similar pox diseases like smallpox and chickenpox. The study employs the Monkeypox 2022 Remastered dataset (Kaggle), which contains both original and augmented images of various pox diseases. Several CNN architectures, including VGG-16, VGG-19, MobileNet V2, GoogLeNet, and EfficientNet-B0, were tested. The highest accuracy was observed for MobileNet V2 (99.25%) on the augmented dataset, while VGG-19 performed best (78.82%) on the original dataset. The research highlights the impact of dataset augmentation on deep learning performance, emphasizing the need for diverse and larger datasets. Future enhancements may include real- time diagnostic tools and improved dataset standardization to enhance model generalizability [4].

The study "Monkeypox Skin Lesion Detection Using Deep Learning Models: A Feasibility Study" by Shams Nafisa Ali, Md. Tazuddin Ahmed, Joydip Paul, Tasnim Jahan,S. M. Sakeef Sani, Nawsabah Noor, and Taufiq Hasan explores AI-based monkeypox detection using deep learning models. The research introduces the Monkeypox Skin Lesion Dataset (MSLD), containing monkeypox, chickenpox, and measles images collected from online sources. Using VGG-16, ResNet50, and

InceptionV3, the study achieves the highest accuracy of 82.96% with ResNet50, followed by VGG-16 (81.48%) and an ensemble model (79.26%). A prototype web application was developed for rapid monkeypox screening. The study emphasizes the necessity of a larger, demographically diverse dataset for enhanced generalizability and aims to refine model performance through improved feature extraction and clinical validation [5].

The study "Reservoir Computing in Epidemiological Forecasting: Predicting Chickenpox Incidence" by Kaushal Kumar applies time-series forecasting to predict chickenpox incidence rates. Utilizing publicly available epidemiological data from Rozemberczki et al., the study compares various deep learning models, including ARIMA, LSTM, Bidirectional LSTM (BLSTM), GRU, Bidirectional GRU (BGRU), and Reservoir Computing. The findings indicate that Reservoir Computing outperforms all other models, achieving the lowest Root Mean Squared Error (RMSE) across multiple test cases. The study highlights the advantages of Reservoir Computing in efficiency and scalability, suggesting further enhancements in model refinement for real-time epidemiological predictions [6].

The study "Deep Learning Models for the Detection of Monkeypox Skin Lesion on Digital Skin Images" by Othman A. Alrusaini explores the application of deep learning (DL) for detecting Monkeypox through digital skin images. The dataset was obtained via web- scraping (Google) using Python’s BeautifulSoup, SERP API, and requests libraries, with images validated by professional physicians. The study utilized multiple CNN architectures, including VGG-16, ResNet50, SqueezeNet, InceptionV3, and SVM classifiers, to classify skin lesions. Among these, VGG-16 achieved the highest accuracy of 96% with an F1-score of 0.92, outperforming other models such as SVM (90%), ResNet50 (90%), SqueezeNet (86%), and InceptionV3 (89%). The research highlights the potential of AI in enhancing Monkeypox diagnosis, reducing diagnostic errors, and aiding in early detection. The study recommends future improvements by expanding datasets and integrating these models into mobile applications for real-time, accessible diagnosis solutions [7].

The study "Explainable Deep Learning Approach for Mpox Skin Lesion Detection with Grad-CAM" by Ghazi Mauer Idroes et al. investigates the use of deep learning models for classifying Mpox skin lesions while incorporating explainable AI techniques like Grad- CAM to enhance interpretability. The dataset consisted of 1,594 images covering six skin conditions (Mpox, chickenpox, cowpox, HFMD, healthy, and measles). The study leveraged ResNet50v2, EfficientNetB4, and DenseNet169 for classification, where ResNet50v2 achieved the highest accuracy of 99.33% with an F1-score of 99.32%, significantly outperforming DenseNet169 (93.94%) and EfficientNetB4 (62.63%). The research demonstrates the robustness of AI-based diagnostic tools in distinguishing Mpox from visually similar diseases and emphasizes the importance of dataset expansion and model improvements to enhance accuracy and reliability [8].

The study "Image Data Collection and Implementation of Deep Learning-Based Model in Detecting Monkeypox Disease Using Modified VGG16" by Md Manjurul Ahsan et al. presents a deep learning approach using a modified VGG16 model for Monkeypox detection. The researchers developed the "Monkeypox2022" dataset, which includes 1,915 images, sourced from open-access platforms and augmented to improve model performance. The study conducted two experimental setups: Study One achieved 97% accuracy (AUC = 97.2), while Study Two attained 88% accuracy (AUC = 86.7). The research highlights the effectiveness of deep learning in medical image classification and suggests further enhancements by expanding the dataset and incorporating explainable AI techniques like LIME to improve model interpretability and trustworthiness [9].

The study "Monkeypox Disease Diagnosis using Machine Learning Approach" by Ajay Krishan Gairola and Vidit Kumar explores the application of machine learning (ML) for diagnosing monkeypox from RGB skin images, addressing the challenges posed by its visual similarity to other pox diseases. Using an open-source dataset of 2,187 images, the study evaluates three convolutional neural networks (CNNs)—AlexNet, GoogleNet, and VGG16— alongside six ML classifiers: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Logistic Regression (LR), Decision Tree (DT), and Naïve Bayes (NB). The VGG16 model with Naïve Bayes and Random Forest classifiers achieved the

highest accuracy of 91.11%, while a fusion-based approach combining CNNs further improved accuracy to 95.55%. The research highlights the potential of ML and deep learning (DL) in automating monkeypox diagnosis, reducing reliance on confirmatory PCR tests, and enabling early detection. Future enhancements include improving dataset diversity, integrating real- time mobile-based diagnostic applications, and refining fusion-based techniques for better generalizability [10].

The study "Early Detection of Monkeypox Skin Disease Using Patch-Based DL Model and Transfer Learning Techniques" by Abbaraju Sai Sathwik et al. investigates deep learning models for classifying monkeypox using a dataset from Kaggle. The authors evaluate multiple CNN architectures, including VGG16, VGG19, ResNet50, ResNet101, and EfficientNet, employing transfer learning techniques. Among these, VGG19 and ResNet50 achieved the highest accuracy of 92%. The study leverages data augmentation techniques to enhance training performance and mitigate the challenge of limited medical image data [11].

The study "Comparison of Monkeypox and Wart DNA Sequences with Deep Learning Model" by Talha Burak Alakus and Muhammet Baykara explores the application of deep learning for distinguishing between monkeypox and wart-causing viruses based on their DNA sequences. By leveraging BiLSTM (Bidirectional Long Short-Term Memory) and various DNA-mapping techniques, the research achieved an accuracy of 96.08% and an F1-score of 99.83%, demonstrating the efficacy of sequence-based classification over visual inspection. The study highlights the potential of bioinformatics and deep learning in disease differentiation, reducing misdiagnosis, and enabling faster, more reliable detection. It also paves the way for future research in expanding datasets, improving classification algorithms, and integrating genomic analysis into automated medical diagnostics [12].

The study "Optimized Deep Learning-Based Monkeypox Diagnostic Framework Using the Metaheuristic Harris Hawks Optimizer Algorithm" by Saleh Ateeq Almutairi proposes a hybrid deep learning and machine learning approach for early-stage monkeypox diagnosis. Using pre-trained CNN models (VGG16, VGG19, Xception, MobileNet, MobileNetV2) optimized with the Harris Hawks Optimizer (HHO) and classified through seven machine learning models, the framework achieved

an accuracy of 97.67% on the MSID dataset and 97.51% on the MPID dataset. The study emphasizes the importance of feature optimization and majority voting in enhancing classification precision, with future work aiming at expanding datasets, refining optimization techniques, and improving real-world deployment [13].

The study "Deep Learning Model for Recognizing Monkeypox Based on DenseNet- 121 Algorithm" by Mohamed Torky, Ali Bakheit, Mohamed Bakry, and Aboul Ella Hassanien investigates the effectiveness of deep learning in diagnosing monkeypox from skin lesion images. The research compares DenseNet-121 and CNN models, with DenseNet-121 achieving a superior testing accuracy of 93%. The study underscores the importance of deep learning in automating infectious disease diagnosis and suggests future improvements through larger datasets and enhanced AI models to improve detection accuracy and reliability [14].

The study "Monkeypox Skin Lesion Detection with MobileNetV2 and VGGNet Models" by Muhammed Coşkun Irmak, Tolga Aydın, and Mete Yağanoğlu explores the application of deep learning models to detect monkeypox skin lesions, addressing the challenge of visual similarity with other diseases like chickenpox and measles. By leveraging pre-trained CNN models, including MobileNetV2, VGG16, and VGG19, the study uses the Kaggle Monkeypox Skin Image Dataset containing 770 images across four classes (Chickenpox, Measles, Monkeypox, and Normal). MobileNetV2 achieved the highest accuracy of 91.37%, followed by VGG16 with 83.62% and VGG19 with 77.58%. Data augmentation was applied to overcome the small dataset size, enhancing model generalization. The study demonstrates the effectiveness of MobileNetV2 in accurate classification and suggests future research could explore other deep learning architectures and ensemble learning techniques to further improve detection performance [15].

The study "A Deep-Learning Algorithm to Classify Skin Lesions from Mpox Virus Infection" by Alexander H. Thieme et al. investigates the use of a custom deep convolutional neural network (MPXV-CNN) for detecting mpox skin lesions, focusing on enhancing early detection and reducing undetected infections. The study assembled a large dataset of 139,198 skin lesion images, with 676 mpox images sourced from scientific literature, social media, and Stanford University Medical

Center, and 138,522 non-mpox images from eight dermatological repositories. The MPXV-CNN achieved a sensitivity of 0.91, specificity of 0.898, and an AUC of 0.966 in the testing cohort. It demonstrated robust classification performance across diverse skin tones and body regions. To facilitate usage, a web-based app was developed. The research underscores the potential of integrating deep learning models into clinical workflows for rapid mpox diagnosis and suggests future advancements could include model explain ability and real-time deployment for outbreak mitigation [16].

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

SYSTEM	64 -bit operating PROCESSOR
RAM	16 GB
HARD DISK	288 GB
KEYBOARD	STANDARD PS/2 KEYBOARD

3.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	WINDOWS 11
SIMULATION TOOLS	Visual Studio
LANGUAGE	Python

3.3 TOOLS DESCRIPTION

- Matplotlib: Used for creating static, animated, and interactive visualizations in Python. It helps visualize training and validation metrics like accuracy and loss.
- Seaborn: Built on top of Matplotlib, Seaborn provides enhanced data visualization features, including heatmaps and correlation matrices .

- Pandas: Imported as pd, pandas is used for efficient data manipulation and analysis. It handles structured data through DataFrames and Series, enabling easy preprocessing and exploration of datasets.

- **TensorFlow/Keras:**

    - tensorflow.keras.layers and tensorflow.keras.models: Used to create neural network architectures with layers like Convolutional, Pooling, Dropout, and Dense layers.

    - tensorflow.keras.preprocessing.image.ImageDataGenerator: Augments and preprocesses image data for better model generalization. It performs real-time data augmentation, such as rotation, flipping, and scaling.

    - tensorflow.keras.applications: Provides pre-trained deep learning models such as ResNet50, DenseNet, InceptionV3, MobileNetV2, VGG19, and NASNet. These models are pre-trained on large datasets like ImageNet and can be fine-tuned for custom classification tasks, such as pox disease classification. Using transfer learning from these models allows faster

    - tensorflow.keras.optimizers.Adam: Optimizer used to improve the model's convergence during training.

    - NumPy: A library for handling numerical operations. It's used for manipulating predictions, labels, and combining features during stacked ensemble implementation.

3.3.1 Python

Python has become one of the most widely used programming languages due to its simplicity, versatility, and extensive library support. Its easy-to-read syntax and vast ecosystem of frameworks make it an ideal choice for developers across various fields, including web development, automation, data science, and artificial intelligence. Python’s cross-platform compatibility and strong community support

further enhance its usability, making it a preferred language for both beginners and professionals.

In the fields of machine learning (ML) and deep learning (DL), Python plays a crucial role due to its comprehensive set of libraries and tools. Libraries such as NumPy and Pandas are used for data preprocessing and manipulation, while Matplotlib and Seaborn help visualize data patterns. For building ML models, Scikit-learn provides a variety of algorithms for classification, regression, and clustering. In deep learning, frameworks like TensorFlow, PyTorch, and Keras allow developers to design and train complex neural networks efficiently. These libraries offer pre-built functions for tasks such as image recognition, natural language processing, and medical diagnostics, making Python a cornerstone in AI research and development.

Python’s extensive support for automation, scalability, and integration with other technologies further solidifies its importance in modern computing. Its ability to handle large datasets, interact with cloud computing services, and deploy AI models in real-world applications makes it a powerful tool across industries. From self-driving cars to medical imaging and financial forecasting, Python continues to drive technological advancements, making it an indispensable language for innovation.

3.3.2 Visual Studio

Visual Studio, developed by Microsoft, is a comprehensive integrated development environment (IDE) that allows users to write, debug, and execute code across various programming languages. It is widely used for software development tasks, including web applications, desktop applications, mobile app development, and game development. Visual Studio provides a powerful and flexible coding environment, suitable for developers, students, and professionals working on a range of projects. With a rich set of features and extensions, Visual Studio simplifies complex software development processes, making it a popular choice in the developer community.

Visual Studio's integration with Azure and GitHub allows developers to deploy applications and manage projects with ease. The platform supports real-time collaboration through Live Share, enabling multiple users to work together on the

same codebase, making it an excellent tool for pair programming and team development.

Additionally, Visual Studio supports various testing frameworks, enabling developers to write, run, and manage tests seamlessly to ensure code quality and reliability.

Visual Studio is available in both free and paid versions. The Community edition provides essential tools for individual developers, open-source projects, and students, while the Professional and Enterprise editions offer advanced features, including performance profiling, enterprise-grade collaboration, and cloud-connected services for larger development teams. With its extensive features, customizable environment, and strong integration with cloud services, Visual Studio is an essential tool for building modern software applications and fostering collaborative development across diverse platforms.

CHAPTER 4

DATASET AND METHODOLOGIES

4.1 DATASET DESCRIPTION

The dataset used in this project is sourced from Kaggle and consists of 10,107 normal images specifically chosen for the classification of pox diseases. This dataset provides a valuable resource for training and evaluating deep learning models in the context of pox disease detection. The dataset includes images categorized into four classes:

- **Chickenpox:** 2,000 images of skin affected by chickenpox, showing lesions caused by the varicella-zoster virus.
- **Cowpox:** 2,000 images displaying cowpox virus infections, characterized by pustular lesions.
- **Monkeypox:** 2,000 images illustrating skin lesions caused by the monkeypox virus, including pustules and rashes.
- **Healthy:** 2,000 images of normal, non-infected skin for comparison.

For model training and evaluation, the dataset was split into training, validation, and testing sets to ensure effective learning and generalization:

- **Training Set:** 8,000 images (2,000 per class).
- **Validation Set:** 2,004 images (500 per class, except Monkeypox with 504).
- **Test Set:** 103 images (16 Chickenpox, 10 Cowpox, 22 Monkeypox, 55 Healthy).

This dataset plays a crucial role in developing a deep learning-based system for automated classification of pox diseases using normal images.

DATASET LINK

https://www.kaggle.com/datasets/dinesh873/poxdata

Figure 4.1 Sample Pox Images

4.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) are deep learning models primarily designed for image classification, object detection, and image segmentation. They consist of convolutional layers, pooling layers, and fully connected layers. Convolutional layers use filters to extract features such as edges, textures, and patterns from input images. Pooling layers reduce spatial dimensions, preserving essential features while reducing computational cost. Activation functions like ReLU introduce non-linearity, enabling the model to learn complex patterns. CNNs automatically learn relevant features, eliminating the need for manual feature

extraction, making them highly effective for visual data analysis.

CNNs are widely used in various fields, including medical imaging, facial recognition, and autonomous vehicles. They achieve high accuracy due to their ability to capture spatial hierarchies in images. However, they require large labeled datasets for training and are computationally intensive, demanding powerful GPUs and substantial memory.

Additionally, CNNs are often considered "black boxes" due to their complex architectures, making it difficult to interpret how they make predictions. Despite these challenges, CNNs remain a fundamental building block in deep learning, influencing the development of advanced architectures.

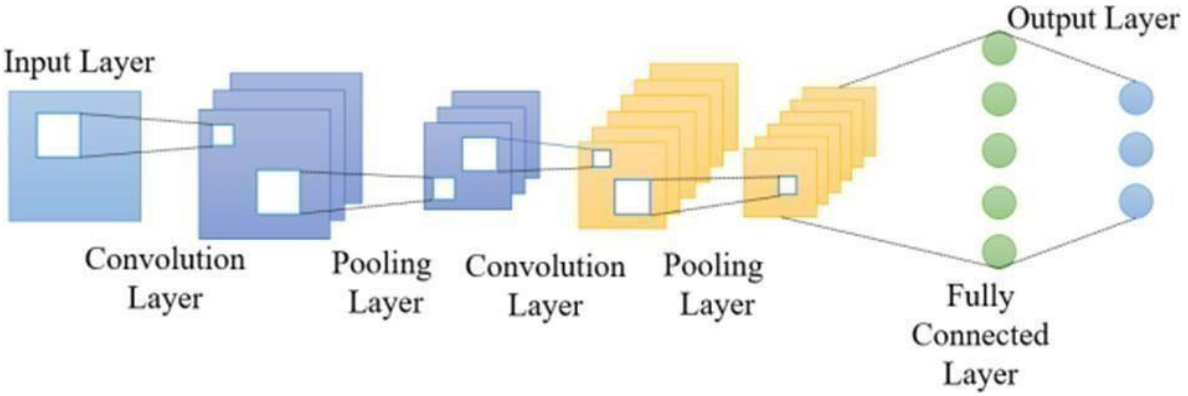

Figure 4.2 CNN Architecture

CNN was used as a baseline deep learning model for pox disease classification. It applied convolutional layers to extract spatial features from images, followed by pooling layers to reduce dimensionality while preserving important patterns. The ReLU activation function was used to introduce non-linearity, preventing vanishing gradient issues. CNN learned low-level to high-level features through multiple convolutional layers. After feature extraction, the flattened output was passed through fully connected layers for classification. The final layer used softmax activation to predict one of the four classes (chickenpox, cowpox, monkeypox, healthy). CNN required **less** computational power compared to deeper architectures but had limited feature extraction capabilities. It was trained using your preprocessed dataset, optimized with cross-entropy loss and an adaptive optimizer. CNN achieved 82.54% accuracy**,** making it a strong starting point for your classification task.

4.3 VGG (Visual Geometry Group)

VGG16 is a deep convolutional neural network architecture developed by the Visual Geometry Group at the University of Oxford. It consists of 16 layers, including 13 convolutional layers and 3 fully connected layers, hence the name VGG16. The model uses small 3×3 convolutional filters consistently throughout its architecture, enabling it to learn complex hierarchical features while maintaining a simple and uniform design. Max pooling layers are used after groups of convolutional layers for down sampling, reducing spatial dimensions while preserving important features. Activation functions like ReLU introduce non-linearity, allowing the network to learn intricate patterns from input images.

VGG16 is known for its simplicity and effectiveness in image classification tasks. It is widely used in transfer learning, where pre-trained models on large datasets like ImageNet are fine-tuned for domain-specific tasks. Despite its deep architecture, VGG16 maintains a straightforward sequential structure, making it easier to implement and modify for various applications. It is popular in computer vision tasks such as object detection, image segmentation, and facial recognition. However, the model's deep architecture also leads to a high number of parameters, resulting in substantial memory usage and long training times. Despite these challenges, VGG16 remains influential in deep learning research due to its balance of simplicity and high accuracy.

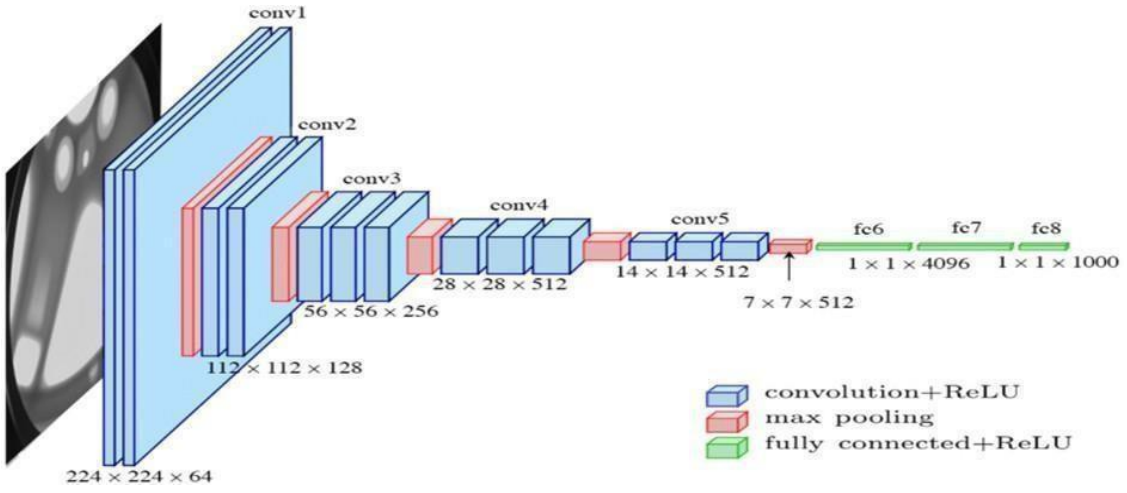

Figure 4.3 VGG16 Architecture

VGG16 is a deep CNN model that stacks multiple small 3×3 convolutional layers instead of using larger filters. It used ReLU activation in each convolutional layer to enable non- linearity and improve feature extraction. Max pooling layers were applied after specific convolutional layers to downsample feature maps and reduce computational cost. VGG's deep architecture allowed it to learn hierarchical features, from edges to complex patterns. The final layers consisted of fully connected layers that classified images using softmax activation**.** Due to its high number of parameters, VGG required more training time and memory compared to CNN. It was trained using cross-entropy loss and an adaptive optimization algorithm to fine- tune weights. Despite being powerful, it achieved 81.55% accuracy, slightly lower than CNN due to overfitting and computational intensity. VGG was a good benchmark but was surpassed by more efficient architectures.

4.4 INCEPTIONV3

InceptionV3 is a state-of-the-art convolutional neural network (CNN) architecture developed by Google, designed to achieve high accuracy in image classification tasks while maintaining computational efficiency. It builds upon the earlier versions of the Inception model by introducing several key optimizations that reduce the number of parameters and improve performance. One of the most significant innovations in InceptionV3 is the use of factorized convolutions, which break down larger convolutional filters into smaller ones (e.g., a 3×3 filter is replaced with two consecutive 1×3 and 3×1 filters). This factorization reduces computational cost and enhances the model's ability to capture complex spatial patterns in images.

Additionally, the model incorporates auxiliary classifiers, which act as additional supervision during training to help the network converge faster and avoid vanishing gradient issues. Another important feature is label smoothing, a regularization technique that prevents the model from becoming overly confident in its predictions, thereby improving generalization and reducing overfitting.

Furthermore, InceptionV3 extensively utilizes batch normalization, which standardizes activations across mini-batches, stabilizing training and accelerating convergence. This technique also minimizes the network's sensitivity to weight initialization, leading to more reliable learning.

The model is also highly versatile, supporting images of various input sizes through the use of global average pooling before the final classification layer. This approach eliminates the need for fully connected layers, reducing the number of parameters and enhancing generalization to different datasets. Due to these optimizations, InceptionV3 is widely used in various computer vision applications, including medical imaging, facial recognition, and autonomous vehicle systems. Its combination of efficiency and accuracy makes it a powerful tool for deep learning practitioners and researchers alike.

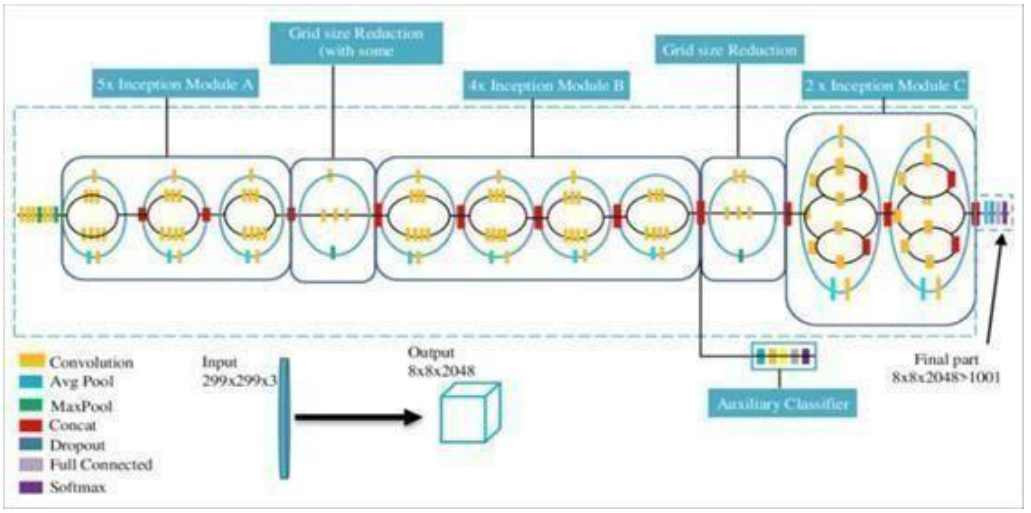

Figure 4.4 InceptionV3 Architecture

InceptionV3 introduced Inception modules, which allowed feature extraction at multiple scales within the same layer. It applied parallel convolutions (1×1, 3×3, 5×5) to capture both fine and coarse details in images. The model used ReLU activation in convolutional layers for non-linearity and batch normalization for stable training. Max pooling helped reduce feature dimensions while maintaining essential information. Unlike VGG, InceptionV3 was designed to be computationally efficient by factorizing convolutions. The final layer used softmax activation to classify images into four categories. The model was trained using cross-entropy loss and Adam optimizer for better convergence. Due to its efficient feature extraction, InceptionV3 outperformed CNN and VGG, achieving 85.43% accuracy. Its deep but optimized architecture made it well-suited for pox disease classification.

4.5 DENSENET

DenseNet (Dense Convolutional Network) is a deep learning architecture designed to improve information flow between layers. Unlike traditional CNNs, where layers learn in isolation, DenseNet connects each layer to every other layer in a feed-forward fashion, maximizing feature reuse and reducing the number of parameters. This dense connectivity improves gradient flow during training, leading to faster convergence and more efficient learning. DenseNet uses batch normalization, ReLU activations, and 1×1 convolutions to reduce the dimensionality of feature maps, maintaining computational efficiency while preserving accuracy.

DenseNet achieves high accuracy with fewer parameters compared to other deep networks, making it memory efficient and less prone to overfitting. It also generalizes well across different datasets, including medical imaging and object detection tasks.

However, the dense connections increase computational complexity, leading to longer training times. Additionally, implementing DenseNet requires careful memory management, especially for very deep networks. Despite these challenges, DenseNet's efficient architecture and superior accuracy have made it a popular choice in deep learning research.

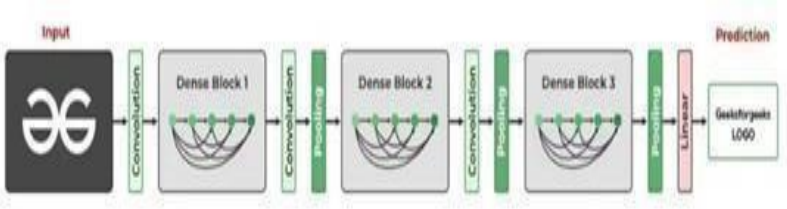

Figure 4.5 DenseNet Architecture

DenseNet used dense connectivity, where each layer received inputs from all previous layers, improving gradient flow and feature reuse**.** Unlike traditional CNNs, it reduced redundancy by sharing features across layers. It applied ReLU activation in convolutional layers to improve learning while avoiding vanishing gradients. Dense

blocks were followed by transition layers with batch normalization and pooling to optimize feature learning. This architecture enabled efficient parameter usage**,** making it less prone to overfitting.

The final layer used softmax activation for classification into four categories. DenseNet was trained using cross-entropy loss and an adaptive optimizer**,** making it robust and accurate. It achieved the highest accuracy of 92.23%, outperforming all other models due to its superior feature propagation. Its success in pox disease classification highlighted the importance of feature reuse and deep learning efficiency**.**

4.6 NASNET (Neural Architecture Search Network)

NASNet is a neural network architecture designed by Google using Neural Architecture Search (NAS)**,** an automated technique to optimize neural network design. It searches for the best convolutional cell structures, which are then stacked to form the complete network.

This approach leads to highly efficient and powerful models with state-of-the-art accuracy on image classification benchmarks. NASNet uses normal cells for feature extraction and reduction cells for down sampling, allowing flexible scaling for different computational resources. Its modular design makes it adaptable to various tasks, including object detection and image segmentation.

NASNet achieves high accuracy with optimized computational efficiency, outperforming many human-designed architectures. It balances performance and resource usage, making it suitable for deployment on mobile devices and embedded systems. However, the search process is computationally expensive, requiring substantial resources and time. Fine-tuning NASNet for specific tasks also demands expert knowledge in neural architecture design. Despite these limitations, NASNet's automated design and state-of-the-art performance have influenced the development of other neural architecture search methods.

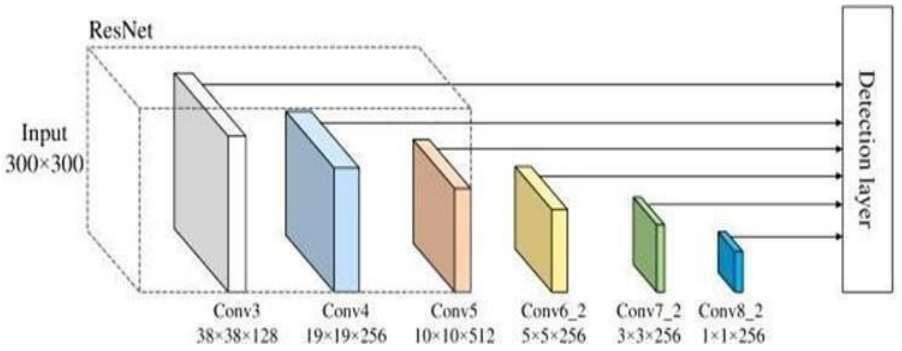

Figure 4.6 NasNet Architecture

NASNet was an automated deep learning architecture optimized using Neural Architecture Search (NAS). It used factorized hierarchical structures to improve feature extraction and adaptively optimized its layer arrangements instead of relying on manual design. The model applied ReLU activation in convolutional layers to introduce non-linearity and enhance learning. It incorporated batch normalization and regularization techniques to improve stability and prevent overfitting. Max pooling and separable convolutions were used to make computation more efficient while maintaining high accuracy. The final fully connected layer utilized softmax activation to classify images into the four disease categories. NASNet was trained using cross-entropy loss and the Adam optimizer, ensuring fast and stable convergence. It achieved 86.41% accuracy, outperforming CNN and VGG but slightly below DenseNet. The strength of NASNet lies in its ability to dynamically search for optimal architectures, making it a promising approach for future improvements in pox disease classification.

The workflow of the proposed model is follows:

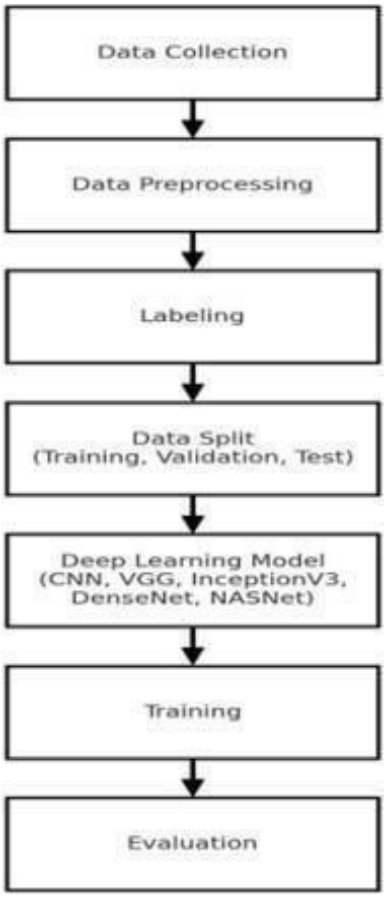

Figure 4.7 Workflow of the proposed model

The diagram represents a Pox Disease Classification Workflow using deep learning models. It involves dataset preprocessing, splitting into training, validation, and testing sets, applying models (CNN, VGG16, InceptionV3, DenseNet, NASNet), and evaluating results using accuracy, precision, recall, specificity, and a confusion matrix.

Explanation for workflow

Data Collection:

● The dataset consists of images of chickenpox, cowpox, monkeypox, and healthy skin, collected from reliable sources. These images are organized into separate folders based on their respective categories to facilitate automatic processing. Ensuring high-quality and diverse images improves the model's ability to generalize across different conditions.

Data Preprocessing:

● All images are resized to match the input size of the model and normalized to ensure consistency in training. Data augmentation techniques like flipping, rotation, and brightness adjustments are optionally applied to improve generalization. These preprocessing steps help in reducing overfitting and enhancing model performance.

Labeling:

● Class labels are assigned automatically based on folder names, ensuring a structured dataset. The labels are converted into a one-hot encoded format, which is required for multi- class classification. Proper labeling prevents data misclassification, which could impact the model’s learning process.

Data Split (test, train):

● The dataset is divided into training, validation, and test sets in a balanced manner. The training set helps the model learn patterns, while the validation set is used for hyperparameter tuning and overfitting prevention. The test set remains unseen during training and is used for final performance evaluation.

Deep Learning Model:

● Pretrained deep learning models like CNN, VGG, InceptionV3, DenseNet, and NASNet are selected for classification. The final output layer is modified to classify images into four categories. Some layers are fine-tuned to learn pox disease-specific features while utilizing the pretrained knowledge of the base models.

Training:

● The model is trained using the training dataset, optimizing parameters such as learning rate and batch size. During training, the validation dataset helps monitor

performance and prevent overfitting. Loss functions like cross-entropy loss and optimizers such as Adam are used for efficient learning.

Evaluation:

- The trained model is tested on the unseen test dataset, and its performance is measured using accuracy, precision, recall, and F1-score. A confusion matrix helps analyze misclassifications and identify model weaknesses. The results are compared across models to determine the most effective architecture for pox disease classification.

CHAPTER 5

RESULTS AND DISCUSSION

CNN-based architectures such as CNN, VGG16, InceptionV3, DenseNet, and NASNet offer unique advantages in medical image analysis. CNN provides a foundational approach with convolutional layers for feature extraction. VGG16, with its simple and deep architecture, is effective in hierarchical feature learning. InceptionV3 utilizes inception modules to optimize computational efficiency while maintaining high classification accuracy. DenseNet121 enhances feature reuse and gradient flow through dense connections, improving model performance. NASNet, an advanced model designed through neural architecture search, balances accuracy and computational efficiency. These models play a critical role in accurately identifying chickenpox, cowpox, monkeypox, and healthy skin conditions from image datasets, aiding in automated diagnosis.

The training phase involves preparing the image dataset by resizing, normalizing, and applying data augmentation techniques to improve model generalization. The dataset is then split into training (80%) and validation (20%) sets to ensure optimal learning. Transfer learning is applied by fine-tuning pre-trained models to extract relevant features from pox disease images. The models are trained using backpropagation, where weights are updated based on cross-entropy loss and optimized using algorithms like Adam or SGD. Early stopping is used to monitor validation loss, preventing overfitting by stopping training when performance declines. To further improve generalization, batch normalization and dropout are integrated into the model.

In the testing phase, the trained models are evaluated using an independent test dataset of unseen images. Each image is processed through the model to classify it into one of four categories: chickenpox, cowpox, monkeypox, or healthy. Performance is assessed using key evaluation metrics, including accuracy, precision, recall, specificity, and F1-score, ensuring a comprehensive analysis of classification reliability. Confusion matrices help identify misclassifications and areas for improvement. Regularization techniques such as dropout and L2 normalization are

applied to enhance model robustness and prevent overfitting. By optimizing these models, the project ensures high accuracy, efficiency, and reliability in detecting pox diseases, making it a valuable tool for medical image classification and automated disease diagnosis.

The research evaluates multiple deep learning models based on key performance metrics, including accuracy, precision, recall, specificity, and F1-score. Among the tested models, DenseNet achieved the highest accuracy of 92.23%, demonstrating strong feature extraction capabilities and computational efficiency. NASNet followed closely with an accuracy of 86.41%, while InceptionV3, CNN, and VGG16 showed promising results in feature extraction and classification but varied in sensitivity and specificity. These findings highlight the effectiveness of deep learning models in pox disease identification, paving the way for further improvements and real-world applications.

Accuracy

Accuracy is found by figuring out how many correct guesses a model makes compared to all the guesses it makes.

$$Acc = \frac{TrPos + TrNeg}{TrPos + TrNeg + FaPos + FaNeg} \quad (5.1)$$

Precision

Precision measures how accurate your positive predictions are. It is found by dividing the number of correctly predicted positive cases by the total number of cases you labeled as positive. This metric evaluates the model's effectiveness in minimizing false positives during classification tasks.

$$Pre = \frac{TrPos}{TrPos + FaPos} \quad (5.2)$$

Recall

Recall is a way to check how well a model can find true positive cases. It measures the percentage of actual positive cases that the model correctly identifies.

This metrics evaluates how effectively a model can detect genuine positives while reducing the occurrence of false negatives.

$$Recal = \frac{TrPos}{TrPos + FaNeg} \quad (5.3)$$

Specificity

Specificity in deep learning (DL) evaluates how effectively a model can identify true negatives, or in other words, its capacity to avoid false positives. This metric is particularly important in fields such as medical diagnostics, where reducing false alarms is critical for accurate results.

$$Specify = \frac{TrNeg}{TrNeg + FaPos} \quad (5.4)$$

F1 Score

The F1 score is a way to measure how well a model performs by combining two important factors: precision and recall. It does this by calculating their harmonic mean, which gives one overall score. This score is especially helpful when you're working with data where some groups are much larger or smaller than others. The F1 score helps balance the importance of precision (how accurate the model is) and recall (how well the model finds all the correct answers).

$$f1 - Score = 2 * \frac{pre * recal}{pre + recal} \quad (5.5)$$

In equation 1,2,3,4, represents an TrPos –TruePositive, TrNeg – TrueNegative, FaPos – FalsePositive, FaNeg – FalseNegative

In our research focused on pox disease identification using deep learning, we evaluated various models including DenseNet, NASNet, InceptionV3, VGG, and CNN using key metrics such as Accuracy, Precision, Recall, Specificity, and F1-score. DenseNet emerged as the top performer with an accuracy of 92.23%, benefiting from its dense connections and ability to propagate features through multiple layers, enabling it to capture complex patterns in the dataset. NASNet

closely followed with an accuracy of 86.41%, showing strong performance but with slight variability across different classes of pox diseases. InceptionV3 achieved an accuracy of 85.43%, demonstrating good generalization but struggling slightly with certain disease categories, such as monkeypox.

VGG performed well with an accuracy of 81.55%, while CNN, though simpler, achieved 82.54%, showing competitive results in terms of accuracy and efficiency.

DenseNet excelled in precision, recall, and F1-score, with an F1-score of 92, demonstrating a strong balance between precision and recall. NASNet followed closely with an F1-score of 87, showing solid overall performance. InceptionV3 and VGG had lower F1- scores of 81 and 85, respectively, indicating challenges in balancing precision and recall. InceptionV3 and NASNet had the highest specificity (above 90%), minimizing false positives, which is crucial for medical disease classification. These results emphasize the need to select models that balance accuracy, precision, recall, and specificity for effective pox disease identification.

Table 5.1 Performance Metrics of the Classifier

Algorithm	Accuracy	Precision	Recall	F1- Score	Specificity
CNN	82.52	85.75	82.55	82.49	78.11
VGG16	81.55	86.98	81.55	81.22	93.44
Inception V3	85.43	81.99	93.18	85.75	93.18
DenseNet	92.23	93.75	92.23	92.34	96.36
NasNet	86.41	85.75	86.55	82.49	95.95

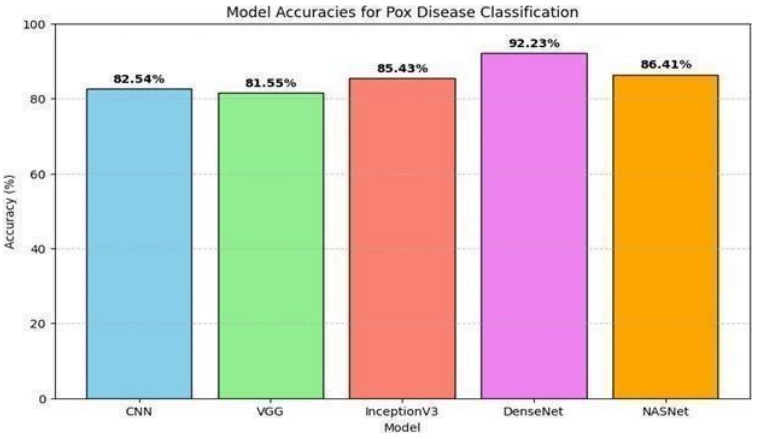

Fig 5.1 Comparison Accuracy for CNN, VGG, Inception V3, DenseNet, NasNet

Table 5.1 and Fig. 5.1 highlights the comparative performance of the models used for pox disease identification, emphasizing DenseNet's superior accuracy and balanced performance across various metrics. This analysis underscores the importance of selecting architectures that effectively balance accuracy, precision, recall, and specificity for optimal performance in medical diagnostic tasks. DenseNet's ability to capture complex patterns and generalize across diverse pox diseases makes it the most suitable model for accurate disease classification, ensuring reliable predictions for clinical applications.

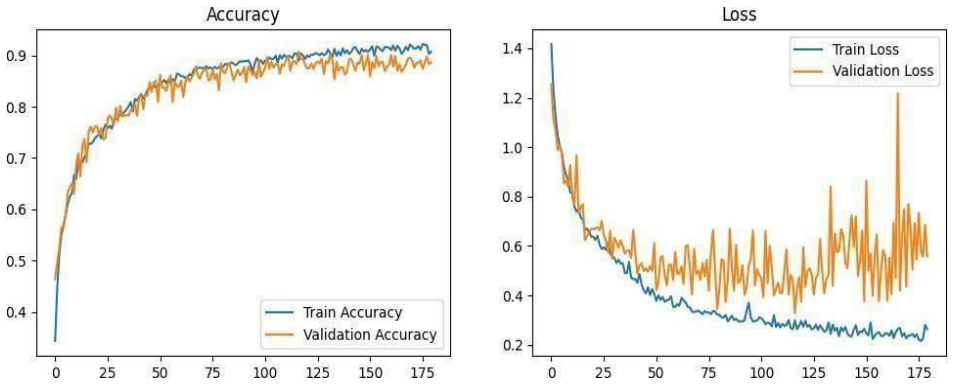

Fig. 5.2. Accuracy and Loss for CNN

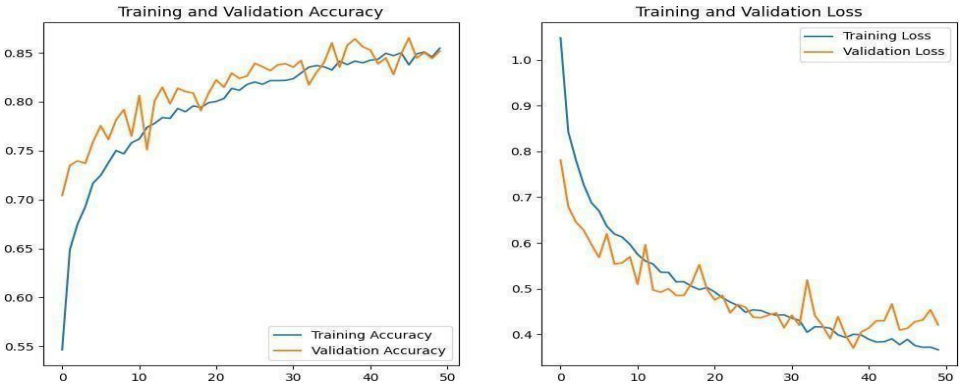

Fig. 5.3. Accuracy and Loss for VGG

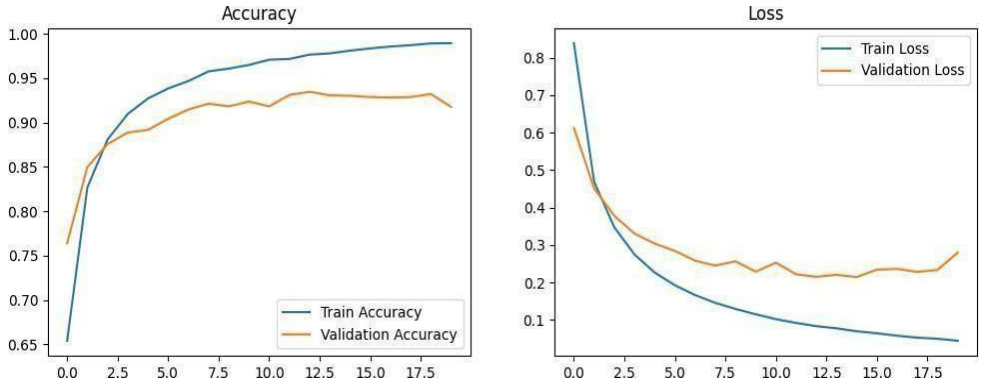

Fig. 5.4. Accuracy and Loss for DenseNet

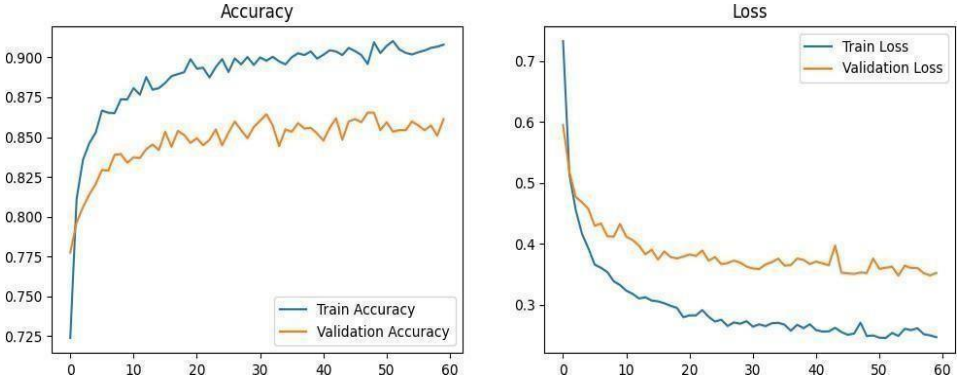

Fig. 5.5. Accuracy and Loss for NasNet

The training and validation accuracy curves show a consistent increase across the epochs, demonstrating effective learning. DenseNet and NASNet achieved the highest validation accuracy, exceeding 90%, highlighting their superior feature extraction capabilities for pox disease classification. In contrast, CNN and VGG16 exhibited slightly lower validation accuracy with noticeable fluctuations, indicating challenges in sensitivity and difficulty in distinguishing true positive cases accurately. These fluctuations suggest that shallower architectures may struggle with complex feature representations, impacting their generalization ability.

The loss curves show a progressively decreasing trend, confirming that the models effectively minimize classification errors over time. Initially, the loss values are high but decline steadily as the models learn from the dataset. However, some models exhibited fluctuations in training loss, particularly those with a high number of parameters, indicating potential overfitting.

DenseNet and NASNet demonstrated the lowest validation loss, reinforcing their ability to generalize better to unseen data. These findings highlight the importance of deeper architectures in achieving improved classification performance for pox disease identification.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The proposed work explored the application of advanced deep learning models, including DenseNet**,** NASNet**,** InceptionV3**,** VGG, and CNN for detecting and classifying pox diseases. The study achieved improved accuracy and reliability in predictions, with DenseNet emerging as the top performer, achieving an accuracy of 92.23%**.** These models demonstrated notable effectiveness, with DenseNet excelling in precision, recall, and F1-score, and NASNet and InceptionV3 showing strong specificity. The findings contribute to the field of medical image classification by introducing robust deep learning models that enhance diagnostic accuracy, ensuring timely and reliable disease detection.

The versatility of these models suggests their potential for broader applications in other medical imaging tasks, such as diagnosing different diseases or conditions, paving the way for wider implementation in healthcare. Future research could focus on further validating these models' generalizability across diverse datasets and expanding their capabilities to address a broader spectrum of medical challenges.

6.2 FUTURE ENHANCEMENTS

In future work, an Ensemble StackedNet will be developed by integrating models like DenseNet, NASNet, and InceptionV3, leveraging their complementary strengths for more accurate predictions. Advanced models such as EfficientNet and Vision Transformers will be explored to enhance accuracy and robustness. Transfer learning and fine-tuning techniques on larger medical datasets will further improve performance. Additionally, explainable AI methods like Grad-CAM will be integrated for better interpretability. A mobile or web application for real-time diagnosis and collaboration with healthcare institutions for clinical validation are also key directions. These advancements aim to improve the accessibility, accuracy, and reliability of automated pox disease diagnosis in real-world settings.

APPENDIX 1

SAMPLE CODING

INCEPTION V3

```
import os

import numpy as np import tensorflow as tf
from tensorflow.keras.preprocessing.image

import ImageDataGenerator from tensorflow.keras.applications import InceptionV3

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D from
tensorflow.keras.models import Model

from sklearn.metrics import classification_report, confusion_matrix, cohen_kappa_score,
accuracy_score, precision_score, recall_score, f1_score

import matplotlib.pyplot as plt import seaborn as sns # Dataset paths
train_dir='D:\MLNadish\Python\dataset\train'
valid_dir='D:\MLNadish\Python\dataset\valid'
test_dir='D:\MLNadish\Python\dataset\test'

# Image Data Generators

train_datagen=ImageDataGenerator(rescale=1./255)

valid_datagen= ImageDataGenerator(rescale=1./255)

test_datagen= ImageDataGenerator(rescale=1./255)

train_gen=train_datagen.flow_from_directory(r'D:\MLProject\python\dataset\train',
target_size=(299, 299),

batch_size=32, class_mode='categorical')


valid_gen = valid_datagen.flow_from_directory( r'D:\ML Project\python\dataset\valid',
target_size=(299, 299),

batch_size=32, class_mode='categorical'

)
```

```
test_gen = test_datagen.flow_from_directory( r'D:\ML Project\python\dataset\test',
target_size=(299, 299),
batch_size=32, class_mode='categorical', shuffle=False
)
# Load InceptionV3 model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(299, 299,
3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(train_gen.num_classes, activation='softmax')(x) model =
Model(inputs=base_model.input, outputs=predictions)
# Freeze base model layers
for layer in base_model.layers:
layer.trainable = False # Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_gen, validation_data=valid_gen, epochs=10)
for layer in base_model.layers[:100]:
layer.trainable = True
# Recompile the model for fine-tuning
model.compile(optimizer=tf.keras.optimizers.Adam(1e-5), loss='categorical_crossentropy',
metrics=['accuracy'])
history_fine = model.fit(train_gen, validation_data=valid_gen, epochs=10)
test_gen.reset()
test_loss, test_accuracy = model.evaluate(test_gen) # Predictions and metrics
predictions = model.predict(test_gen) y_pred = np.argmax(predictions, axis=1) y_true =
test_gen.classes
class_labels = list(test_gen.class_indices.keys()) # Classification report
```

report = classification_report(y_true, y_pred, target_names=class_labels) print(report) #

Confusion matrix

cm = confusion_matrix(y_true, y_pred) plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)

plt.title('Confusion Matrix') plt.xlabel('Predicted') plt.ylabel('True') plt.show()
# Calculate metrics

accuracy = accuracy_score(y_true, y_pred) kappa = cohen_kappa_score(y_true, y_pred)

precision = precision_score(y_true, y_pred, average='macro') recall = recall_score(y_true, y_pred, average='macro')

f1 = f1_score(y_true, y_pred, average='macro') specificity = np.mean(np.diag(cm) / np.sum(cm, axis=1)) error_rate = 1 – accuracy

print(f"Accuracy: {accuracy}") print(f"Kappa: {kappa}") print(f"Precision: {precision}") print(f"Recall: {recall}") print(f"F1 Score: {f1}") print(f"Specificity: {specificity}") print(f"Error Rate: {error_rate}")

Plot training and validation accuracy without epochs plt.figure(figsize=(12, 6)) plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val_accuracy'], label='Validation Accuracy') plt.plot(history_fine.history['accuracy'], label='Fine-Tune Train Accuracy') plt.plot(history_fine.history['val_accuracy'], label='Fine-Tune Validation Accuracy')

plt.legend()

plt.title('Training and Validation Accuracy') plt.xlabel('Iterations') plt.ylabel('Accuracy') plt.xticks([])

plt.show()

DENSENET

import os

import numpy as np import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator from

```
tensorflow.keras.applications import DenseNet121

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D from
tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from sklearn.metrics import classification_report, confusion_matrix, cohen_kappa_score,
accuracy_score

import matplotlib.pyplot as plt import seaborn as sns # Set dataset paths

train_dir = 'D:\DL PROJECT\python\dataset\train' valid_dir = 'D:\DL
PROJECT\python\dataset\valid' test_dir = 'D:\DL PROJECT\Python\dataset\test'

# Image dimensions and batch size img_size = (224, 224) batch_size = 32

# Data Augmentation and Image Generators train_datagen =
ImageDataGenerator(rescale=1./255) valid_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255) train_generator =
train_datagen.flow_from_directory(

r'D:\DL PROJECT\Python\dataset\train', target_size=img_size, batch_size=batch_size,
class_mode='categorical'

)

valid_generator = valid_datagen.flow_from_directory(

    r'D:\DL PROJECT\Python\dataset\valid', target_size=img_size, batch_size=batch_size,
class_mode='categorical'

)

test_generator = test_datagen.flow_from_directory(

r'D:\DL PROJECT\Python\dataset\test', target_size=img_size, batch_size=batch_size,
class_mode='categorical', shuffle=False

)

# Load DenseNet121 Model

base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(224,
224, 3)) x = GlobalAveragePooling2D()(base_model.output)

x = Dense(128, activation='relu')(x) x = Dense(64, activation='relu')(x)
```

```
output_layer = Dense(4, activation='softmax')(x)  # 4 classes model =
Model(inputs=base_model.input, outputs=output_layer) # Freeze the base model layers
for layer in base_model.layers: layer.trainable = False # Compile Model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
# Train Model
history = model.fit(train_generator, validation_data=valid_generator, epochs=20) #
Evaluate Model
test_generator.reset()
y_true = test_generator.classes
y_pred_probs = model.predict(test_generator) y_pred = np.argmax(y_pred_probs, axis=1) #
Classification Report
report = classification_report(y_true, y_pred,
target_names=test_generator.class_indices.keys(), output_dict=True)
accuracy = accuracy_score(y_true, y_pred) kappa = cohen_kappa_score(y_true, y_pred)
precision = report['weighted avg']['precision'] recall = report['weighted avg']['recall']
f1_score = report['weighted avg']['f1-score']
specificity = np.mean([report[label]['recall'] for label in
test_generator.class_indices.keys()]) error_rate = 1 - accuracy
print(f"Accuracy: {accuracy:.4f}") print(f"Cohen's Kappa: {kappa:.4f}") print(f"Precision:
{precision:.4f}") print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1_score:.4f}") print(f"Specificity: {specificity:.4f}") print(f"Error Rate:
{error_rate:.4f}") # Confusion Matrix
cm = confusion_matrix(y_true, y_pred) # Plot Training Accuracy & Loss
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
```

```
plt.title('Accuracy') plt.subplot(1, 2, 2) plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss') plt.legend()

plt.title('Loss') plt.show()

# Plot Confusion Matrix plt.figure(figsize=(6, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=test_generator.class_indices.keys(),
yticklabels=test_generator.class_indices.keys())

plt.title('Confusion Matrix')
plt.xlabel('Predicted')

plt.ylabel('True')

 plt.show()
```

CNN

```
import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator from
tensorflow.keras import layers, models

import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,
precision_score, recall_score, f1_score, cohen_kappa_score

# Paths to the datasets (update these)

train_dir = 'D:\DL PROJECT\python\dataset\train' valid_dir = 'D:\DL
PROJECT\python\dataset\valid' test_dir = 'D:\DL PROJECT\Python\dataset\test'
# Image size and batch size img_height = 224

img_width = 224

batch_size = 32

# Data generators for preprocessing and augmentation train_datagen =
ImageDataGenerator(

rescale=1./255, rotation_range=40, width_shift_range=0.2, height_shift_range=0.2,
```

```
shear_range=0.2, zoom_range=0.2, horizontal_flip=True, fill_mode='nearest')
valid_datagen = ImageDataGenerator(rescale=1./255) test_datagen =
ImageDataGenerator(rescale=1./255)  # Load images from directories
train_generator = train_datagen.flow_from_directory(
   r'D:\DL PROJECT\Python\dataset\train', target_size=(img_height, img_width),
batch_size=batch_size, class_mode='categorical')
valid_generator = valid_datagen.flow_from_directory(
   r'D:\DL PROJECT\Python\dataset\valid', target_size=(img_height, img_width),
batch_size=batch_size, class_mode='categorical')
test_generator = test_datagen.flow_from_directory(
   r'D:\DL PROJECT\Python\dataset\test', target_size=(img_height, img_width),
batch_size=batch_size, class_mode='categorical', shuffle=False)
# Build CNN model
model = models.Sequential([
layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
layers.MaxPooling2D((2, 2)),
layers.Conv2D(64, (3, 3), activation='relu'),
layers.MaxPooling2D((2, 2)),
layers.Conv2D(128, (3, 3), activation='relu'),
layers.MaxPooling2D((2, 2)), layers.Flatten(),
layers.Dense(512, activation='relu'), layers.Dropout(0.5),
   layers.Dense(4, activation='softmax') # 4 classes: chickenpox, cowpox, healthy,
monkeypox
])
# Compile the model model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# Train the model history = model.fit(
train_generator, epochs=180,
    validation_data=valid_generator) # Evaluate on test set
test_loss, test_acc = model.evaluate(test_generator) print(f'Test Accuracy: {test_acc:.4f}')
# Get true labels and predictions y_true = test_generator.classes
y_pred_probs = model.predict(test_generator) y_pred = np.argmax(y_pred_probs, axis=1)
# Compute evaluation metrics
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred, average='weighted') recall =
recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted') kappa = cohen_kappa_score(y_true,
y_pred) error_rate = 1 - accuracy
# Specificity Calculation
cm = confusion_matrix(y_true, y_pred)
specificity_list = [cm[i][i] / sum(cm[:, i]) if sum(cm[:, i]) != 0 else 0 for i in range(len(cm))]
specificity = np.mean(specificity_list)
# Print metrics
print(f"Accuracy: {accuracy:.4f}") print(f"Precision: {precision:.4f}") print(f"Recall:
{recall:.4f}")
print(f"F1 Score: {f1:.4f}") print(f"Specificity: {specificity:.4f}") print(f"Cohen's Kappa:
{kappa:.4f}") print(f"Error Rate: {error_rate:.4f}") # Plot Training Accuracy & Loss
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy') plt.legend()
plt.title('Accuracy') plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss') plt.plot(history.history['val_loss'],
```

```
label='Validation Loss') plt.legend()
plt.title('Loss') plt.show()

# Plot Confusion Matrix
plt.figure(figsize=(6, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=test_generator.class_indices.keys(),
yticklabels=test_generator.class_indices.keys())

plt.title('Confusion Matrix') plt.xlabel('Predicted') plt.ylabel('True')
plt.show()

# Save model model.save('pox_classifier_model1.h5')
```

VGG

```
# Import necessary libraries import os
import numpy as np import tensorflow as tf

from sklearn.metrics import confusion_matrix, classification_report, cohen_kappa_score,
precision_score, recall_score, f1_score

import matplotlib.pyplot as plt import seaborn as sns

from tensorflow.keras.preprocessing.image import ImageDataGenerator from
tensorflow.keras.applications import VGG16

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten, Dropout from
tensorflow.keras.optimizers import Adam

# Define paths for datasets

train_dir = 'D:\DL PROJECT\python\dataset\train' valid_dir = 'D:\DL
PROJECT\python\dataset\valid'

test_dir = 'D:\DL PROJECT\Python\dataset\test' # Parameters
image_size = (224, 224)

batch_size = 32

num_classes = 4

epochs = 50
```

```
    # Data generators
train_datagen = ImageDataGenerator( rescale=1.0 / 255, rotation_range=20,
    width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True
    )
    valid_datagen = ImageDataGenerator(rescale=1.0 / 255) test_datagen =
    ImageDataGenerator(rescale=1.0 / 255) train_generator =
    train_datagen.flow_from_directory(
    r'D:\DL PROJECT\Python\dataset\train', target_size=image_size, batch_size=batch_size,
    class_mode='categorical'
    )
valid_generator = valid_datagen.flow_from_directory( r'D:\DL
    PROJECT\Python\dataset\valid',
   target_size=image_size, batch_size=batch_size, class_mode='categorical'
    )
test_generator = test_datagen.flow_from_directory( r'D:\DL PROJECT\Python\dataset\test',
    target_size=image_size,
    batch_size=batch_size, class_mode='categorical', shuffle=False
    )
    # Load pre-trained VGG16 model
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    # Freeze the base model
for layer in base_model.layers: layer.trainable = False
    # Add custom classification layers x = base_model.output
    x = Flatten()(x)
    x = Dense(256, activation='relu')(x) x = Dropout(0.5)(x)
    output = Dense(num_classes, activation='softmax')(x) # Create the model
```

```
model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])
# Train the model history = model.fit(
train_generator, epochs=epochs,
validation_data=valid_generator
)
# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_generator) print(f"Test accuracy: {test_acc}")
# Save the model model.save('vgg16_finetuned_model.h5') # Predict on the test set
predictions = model.predict(test_generator) predicted_classes = tf.argmax(predictions,
axis=1).numpy() true_classes = test_generator.classes
class_labels = list(test_generator.class_indices.keys()) # Confusion matrix
conf_matrix = confusion_matrix(true_classes, predicted_classes) print("Confusion
Matrix:\n", conf_matrix)
# Plot Confusion Matrix
def plot_confusion_matrix(conf_matrix, class_labels): plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels,
yticklabels=class_labels)
plt.xlabel('Predicted') plt.ylabel('True') plt.title('Confusion Matrix') plt.show()
plot_confusion_matrix(conf_matrix, class_labels) # Calculate metrics
report = classification_report(true_classes, predicted_classes, target_names=class_labels,
output_dict=True)
precision = precision_score(true_classes, predicted_classes, average='weighted') recall =
recall_score(true_classes, predicted_classes, average='weighted')
f1 = f1_score(true_classes, predicted_classes, average='weighted') kappa =
cohen_kappa_score(true_classes, predicted_classes) error_rate = 1 - test_acc
# Specificity calculation specificity_per_class = []
```

```
    for i in range(len(class_labels)):
        tn = np.sum(conf_matrix) - (np.sum(conf_matrix[i, :]) + np.sum(conf_matrix[:, i]) -
    conf_matrix[i, i])
    fp = np.sum(conf_matrix[:, i]) - conf_matrix[i, i] specificity_per_class.append(tn / (tn +
    fp))
    specificity = np.mean(specificity_per_class) # Print metrics
    print("Classification Report:")
for label, metrics in report.items(): if label in class_labels:
            print(f"{label}: {metrics}") print(f"Precision: {precision}")
    print(f"Recall: {recall}") print(f"F1 Score: {f1}") print(f"Cohen's Kappa: {kappa}")
    print(f"Error Rate: {error_rate}") print(f"Specificity: {specificity}") # Plot accuracy and
    loss
def plot_training_history(history): acc = history.history['accuracy']
    val_acc = history.history['val_accuracy'] loss = history.history['loss']
    val_loss = history.history['val_loss'] epochs_range = range(len(acc)) plt.figure(figsize=(12,
    6))
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy') plt.plot(epochs_range, val_acc,
    label='Validation Accuracy') plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy') plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss') plt.plot(epochs_range, val_loss,
    label='Validation Loss') plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')
    plt.show()
```

APPENDIX 2

SCREENSHOTS

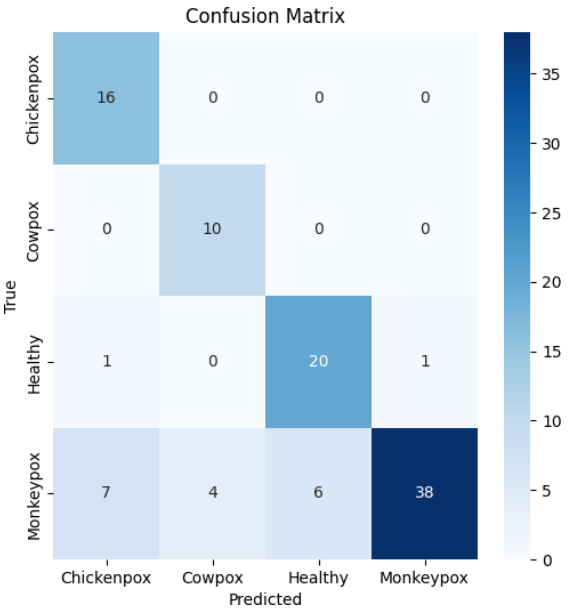

Fig. A.2.1 Confusion Matrix for CNN

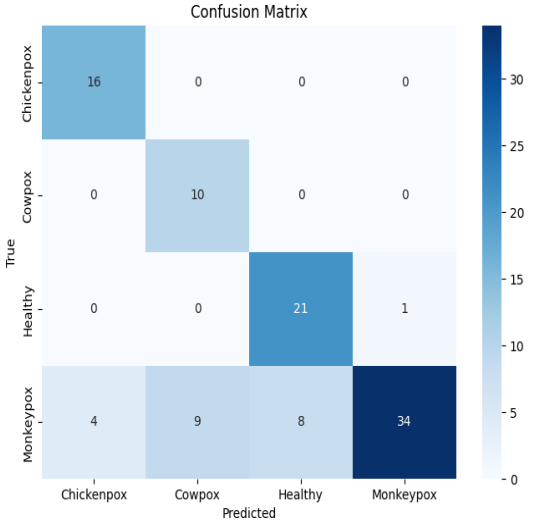

Fig. A.2.2 Confusion Matrix for VGG

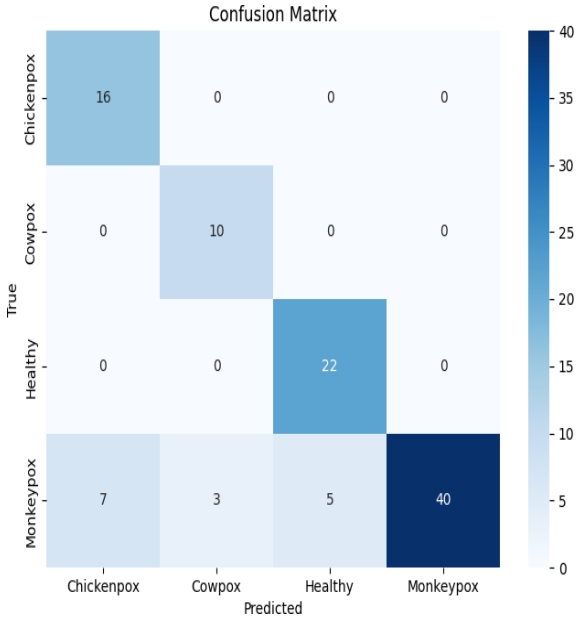

Fig. A.2.3 Confusion Matrix for Inception V3

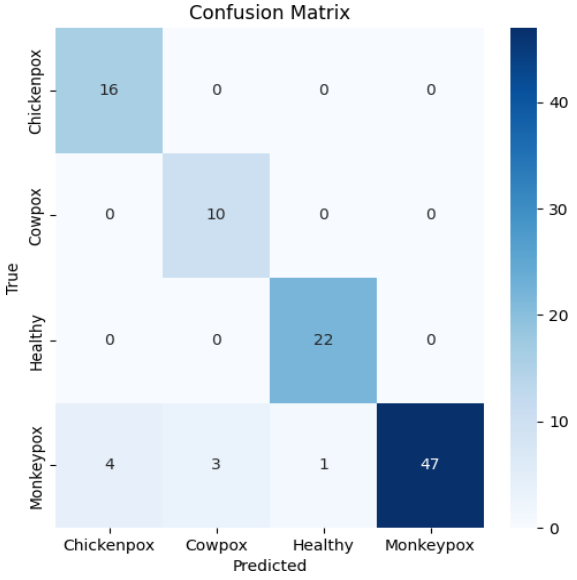

Fig. A.2.4 Confusion Matrix for DenseNet

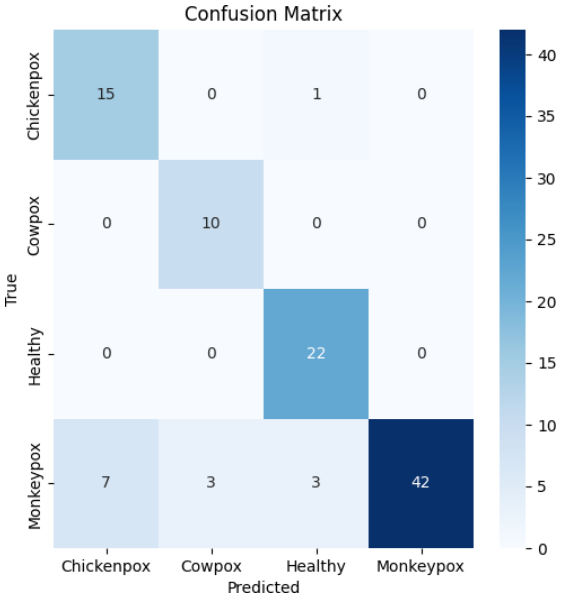

Fig. A.2.5 Confusion Matrix for NasNet

REFERENCES

[1] D. Uzun Ozsahin, M. T. Mustapha, B. Uzun, B. Duwa, and I. Ozsahin, "Computer-Aided Detection and Classification of Monkeypox and Chickenpox Lesion in Human Subjects Using Deep Learning Framework," Diagnostics, vol. 13, no. 292, pp. 1–14, Jan. 2023. DOI: 10.3390/diagnostics13020292.

[2] M. Pal et al., "Deep and Transfer Learning Approaches for Automated Early Detection of Monkeypox (Mpox) Alongside Other Similar Skin Lesions and Their Classification," ACS Omega, vol. 8, no. 35, pp. 31747–31757, Aug. 2023. DOI: 10.1021/acsomega.3c02784.

[3] C. Sitaula and T. B. Shahi, "Monkeypox Virus Detection Using Pre-trained Deep Learning-based Approaches," Journal of Medical Systems, vol. 46, no. 78, pp. 1–9, Oct. 2022. DOI: 10.1007/s10916-022-01868-2.

[4] M. Çelik and Ö. İnİk, "Detection of Monkeypox Among Different Pox Diseases with Different Pre-Trained Deep Learning Models," Journal of the Institute of Science and Technology, vol. 13, no. 1, pp. 10–21, Jan. 2023. DOI: 10.21597/jist.1206453.

[5] S. N. Ali, M. T. Ahmed, J. Paul, T. Jahan, S. M. S. Sani, N. Noor, and T. Hasan, "Monkeypox Skin Lesion Detection Using Deep Learning Models: A Feasibility Study," arXiv preprint, Jul. 2022. Available: arXiv:2207.03342.

[6] K. Kumar, "Reservoir Computing in Epidemiological Forecasting: Predicting Chicken Pox Incidence," medRxiv preprint, Apr. 2023. DOI: 10.1101/2023.04.24.23289018.

[7] O. A. Alrusaini, "Deep Learning Models for the Detection of Monkeypox Skin Lesion on Digital Skin Images," International Journal of Advanced Computer Science and Applications, vol. 14, no. 1, pp. 637–645, Jan. 2023. Available: IJACSA.

[8] G. M. Idroes, T. R. Noviandy, T. B. Emran, and R. Idroes, "Explainable Deep Learning Approach for Mpox Skin Lesion Detection with Grad-CAM," Heca Journal of Applied Sciences, vol. 2, no. 2, pp. 54–65, Sep. 2024. DOI:

10.60084/hjas.v2i2.216.

[9] M. M. Ahsan, M. R. Uddin, M. Farjana, A. N. Sakib, K. A. Momin, and S. A. Luna, "Image Data Collection and Implementation of Deep Learning-Based Model in Detecting Monkeypox Disease Using Modified VGG16," arXiv preprint, Jun. 2022. Available: arXiv:2206.01862.

[10] A. K. Gairola and V. Kumar, "Monkeypox Disease Diagnosis Using Machine Learning Approach," 2022 IEEE International Conference on Smart Computing (ICSC), Nov. 2022. DOI: 10.1109/ICSC56524.2022.10009135.

[11] A. S. Sathwik, B. Naseeba, J. C. Kiran, K. Lokesh, V. S. D. Ch, and N. P. Challa, "Early Detection of Monkeypox Skin Disease Using Patch Based DL Model and Transfer Learning Techniques," EAI Endorsed Transactions on Pervasive Health and Technology, vol. 9, Nov. 2023. DOI: 10.4108/eetpht.9.4313.

[12] T. B. Alakus and M. Baykara, "Comparison of Monkeypox and Wart DNA Sequences with Deep Learning Model," Applied Sciences, vol. 12, no. 20, Oct. 2022. DOI: 10.3390/app122010216.

[13] S. A. Almutairi, "DL-MDF-OH2: Optimized Deep Learning-Based Monkeypox Diagnostic Framework Using the Metaheuristic Harris Hawks Optimizer Algorithm," Electronics, vol. 11, no. 4077, Dec. 2022. DOI: 10.3390/electronics11244077.

[14] M. Torky, A. Bakheit, M. Bakry, and A. E. Hassanien, "Deep Learning Model for Recognizing Monkeypox Based on DenseNet-121 Algorithm," medRxiv preprint, Dec. 2022. DOI: 10.1101/2022.12.20.22283747.

[15] M. C. Irmak, T. Aydın, and M. Yağanoğlu, "Monkeypox Skin Lesion Detection with MobileNetV2 and VGGNet Models," 2022 IEEE International Conference, Oct.–Nov. 2022. DOI: 10.1109/ICSC56524.2022.10009135.

[16] A. H. Thieme et al., "A Deep-Learning Algorithm to Classify Skin Lesions from Mpox Virus Infection," Nature Medicine, vol. 29, pp. 738–747, Mar. 2023. DOI: 10.1038/s41591-023-02225-7.

[17] S. H. Khan, R. Iqbal, and S. Naz, "A Recent Survey of the Advancements in Deep Learning Techniques for Monkeypox Disease Detection," Artificial Intelligence Lab, University of Engineering and Applied Sciences, 2023. Available: Research Article.

[18] S. Savaş, "Enhancing Disease Classification with Deep Learning: A Two-Stage Optimization Approach for Monkeypox and Similar Skin Lesion Diseases," Journal of Imaging Informatics in Medicine, vol. 37, pp. 778–800, Jan. 2024. DOI: 10.1007/s10278-023- 00941-7.

[19] M. M. Ahsan, M. S. Ali, M. M. Hassan, T. A. Abdullah, K. D. Gupta, U. Bagci, C. Kaushal, and N. F. Soliman, "Monkeypox Diagnosis With Interpretable Deep Learning," IEEE Access, vol. 11, pp. 81965–81977, Aug. 2023. DOI: 10.1109/ACCESS.2023.3300793.

[20] A. K. Mandal, "Usage of Particle Swarm Optimization in Digital Images Selection for Monkeypox Virus Prediction and Diagnosis," Research Square preprint, Jan. 2023. DOI: 10.21203/rs.3.rs-2421266/v1.