Scientiarum: A Multidisciplinary Journal
Volume 2, Issue 2, April 2026, pp. 1–18
DOI: 10.54646/SAPARS.2026.30

SAPAR SERVICES LLP
Scholarly Articles Publication And Repository Services

RESEARCH

Transfer learning methods for classifying medical images and identifying illnesses with minimal annotated data

Manikanta Akula[1]* and Deekshitha Kadiyala[2]

[1]Department of Information Systems, University of Memphis, Memphis, TN, United States
[2]Department of Computer Science, University of Memphis, Memphis, TN, United States

***Correspondence:**
Manikanta Akula,
manikantaakula78@gmail.com

Received: 27 March 2026; **Accepted:** 10 April 2026; **Published:** 23 April 2026

Medical image classification is often constrained by the unavailability of large, annotated data-sets owing to the unreasonable cost of expert labeling and class discrepancy in rare conditions. One of the interesting techniques to overcome these drawbacks is the procedure of transfer learning (TL) which utilizes the knowledge gained from deep learning (DL) models that were formerly trained. DL has achieved significant success in feature representation from complex data, but it generally involves a considerable amount of annotated information for effective training. Using small and imbalanced data-sets, this paper investigates the performance of TL for medical image classification tasks using VGG16, ResNet50, and EfficientNet. A scratch-built baseline Convolutional Neural Network (CNN) is adopted as a reference to compare TL models. COVID-19 chest X-ray databases are utilized in experiments. Accuracy, F1-score, and training efficiency are applied to measure performance. While TL has illustrated impressive performance, often similar to or exceeding diagnosis, its configuration in numerous studies has been defined as arbitrary. Previous studies have revealed that TL is better than scratch-built models on tiny medical datasets, resulting in a COVID-19 detection precision of $>96\%$ using ResNet50. Also pointed out the usefulness of ensemble TL approaches. This paper shows that TL is a practicable and effective means to expand medical image solutions in cases with limited data. That makes it probable to quickly deploy artificial intelligence (AI) tools into real-world healthcare applications.

Keywords: deep learning, convolutional neural networks, VGG 16, EfficientNet, COVID-19 chest X-rays, ResNes50, imbalanced data, feature extraction

Introduction

In the area of medical image analysis, deep learning (DL) techniques have attained remarkable success in learning complex and task-specific feature representation from intricate data, allowing impressive outcomes in disease diagnosis (1, 2). Convolutional Neural Networks (CNNs), in particular, have demonstrated their capacity to learn intricate patterns from scans like magnetic resonance imaging (MRI) and X-ray images, generating them a powerful utility for automated disease detection (3). However, a significant and continual bottleneck for these DL processes is their substantial reliance on large, high-quality annotated data-sets for effective training. This challenge is additional compounded by the intrinsic scarcity and class imbalance of information for uncommon conditions (4). To address this crucial data scarcity problem, the process of transfer learning (TL) has emerged as a highly effective and generally accepted method in medical imaging (5, 6). TL operates on the theory of leveraging knowledge gained from a DL model that has been formerly trained on a large and diverse source data-set, such as ImageNet, and using that knowledge to a new, but related, target task in this case, medical image classification (6, 7). For medical imaging applications, the transfer of information from ImageNet has become a de-facto approach, despite the notable deviations in tasks and image characteristics between the domains. The effectiveness of TL has been illustrated across a extensive range of

medical imaging applications (8). Similarly, in response to the international health crisis, TL has played a pivotal role in the fast and automatic finding of the innovative corona virus disease 2019 (COVID-19) from chest X-ray images (9, 10). This capability of these models to rapidly and precisely classify disease from medical scans has proven to be a viable method for detecting multiple abnormalities even with tiny medical data-sets, often yielding impressive results (10, 11).

However, despite its widespread adoption and proven benefits, the role of TL in medical imaging is not without its challenges and areas for additional research. The factors that ascertain whether and to what extent TL to the medical domain is advantageous remain unclear, and the long-standing hypothesis that features from source domains are directly reusable for medical images has lately been called into question (12).

This study seeks to provide a thorough review of the purpose of TL in medical image classification, concentrating on its efficacy in addressing the difficulty of data scarcity. A core baseline model, a scratch-built CNN, will be used as a reference point for comparing the performance of TL models, as found in some of the appraised studies. The performance metrics, such as accuracy, F1-score, and training efficiency, will be considered to estimate the suitability of the TL approach (13).

The core approach for TL comprises using pre-trained CNN models, such as VGG, ResNet, INceptionV3, MobileNet, and Xception, as feature extractors (14). This fine-tuning procedure allows the model to adapt the high-level features learned from the source domain to the nuances of the medical images while retaining the low-level features extraction capabilities. To further alleviate the tiny sample size drawback inherent in medical imaging, data augmentation methods are commonly used to the training information to increase the size and diversity of the data-set (15).

This study mainly focuses on exploring the usage of TL. This technique can be used to apply knowledge from models predefined on large data-sets. In this work, comparing a systematic experiments as a small CNN trained from scratch to several transfer-learning techniques, such as (a) a frozen feature extractor with a new classifier head, (b) full fine-tuning, and (c) partial fine-tuning of higher layers.

Medical imaging data-sets that are used for experimentation is:

COVID-19 Chest X-ray images: These are used to classify infected versus non-infected cases.

This study evaluates the performance of widely used models such as VGG16, ResNet50 and EfficientNet, comparing them against the baseline of a CNN model trained from scratch. The assessment is measured with specific performance metrics. It mainly insists that TL must achieve outcomes with scalability, speed and accuracy rather than poor results. It will enable faster deployment of artificial intelligence (AI) tools in the healthcare setting, where big data is very challenging to achieve.

Literature review

Many researchers are utilizing DL approaches to learn highly intricate pattern. It has also been observed that an group of predictions from many models gives better performance than that of a single model. However, two major bottlenecks for developing ensemble DL models are their excessive computational complexity and the establishment of a large sample size for better generalization. It effectively tells the patient about disease treatment, health tracking, and continuously monitoring. With the use of annotated data-sets that are limited, even though they effectively benefit DL models. It enables researchers to assess TL as a practical solution that uses knowledge from a pre-trained model with specified conditions on large data-sets like ImageNet (**Figure 1**).

The application of DL in medical imaging has significant advantages, but it is limited by annotated data-sets. It highlights the expert-labeled data and presence of rare conditions that can hinder the effectiveness of models trained from scratch. It developed a DL platform with the advantage of optical coherence tomography (OCT) images that automatically helps to identify medical conditions. Demonstrate the CNN potential in dermatology and their reliance on large data-sets that may replicate difficult domains such as radiology. To handle these issues, researchers have turned to TL as a practical solution (16).

It explains the importance of CNNs in imaging. Emphasizes how CNNs are superior to traditional machine learning tasks in image classification, segmentation, etc. For the past few years, researchers have been using the advantage of CNN-based models to extract and gather unique features for the diagnosis of various diseases. Using a convolutional network-based model achieved a good level of accuracy in comparison with traditional machine learning and many other techniques that are manually performed by physicians (17).

It explains about the first segmented the MRI image into two segments: gray and white matter. They used a multiscale ConvNet for the diagnosis of Alzheimer's Disease (AD). In terms of accuracy, MSCNet has a higher performance level than ResNet-50 in the NC and MCI classes of matter (gray and white) is about 98.85% and 98.11% and ResNet accuracy is about 96.01% and 95.88%. This study shows that lower computational power and fewer parameters, the CNN- based MSCNet model performs well for the medical image data-set (18) (**Table 1**).

From the computational perspective, using TL with a CNN model tends to significantly reduce the time and resources that are needed to train a CNN model from scratch. It also improves the network performance. It is because of the

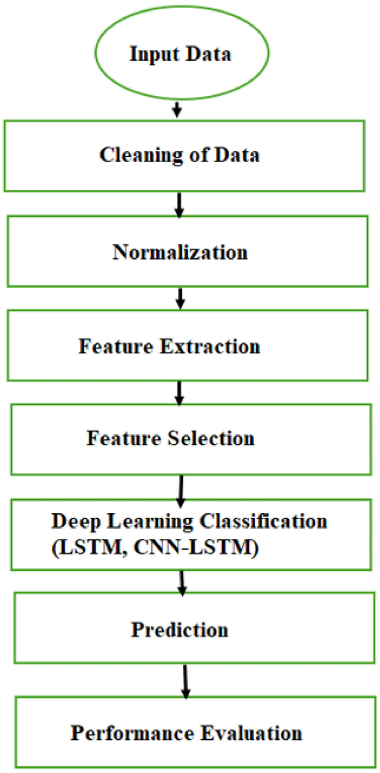

FIGURE 1 | Work-flow.

presence of TL allows us to efficiently transfer knowledge from pre-trained models to our new tasks. It results in less usage of data and increases computation with good levels of accuracy. It mainly highlights how CNN automatically extracts advanced features through the effective usage, and supports CNN baseline and CNN TL in our experiment. The results from the experimentation where CNNs and TL techniques are involved are referred to as "black-box" it is because features learned by the network are very difficult to interpret by medical experts. Its advantage is the limited availability of annotated image data with accuracy (19). High-performing CNNs typically require large annotated data-sets. However, class imbalance is a natural problem for rare diseases, and medical data annotation is costly and requires subject-matter expertise. It has the major drawback for developing and deploying AI models in healthcare because of its high cost and time that are needed for data collection and expert annotation, the privacy concerns that surround sensitive patient information and the limited prevalence of rare diseases (20).

The major reasons for the scarcity will be:

i High annotation and acquisition costs.

ii Overuse of the imaging procedures due to the potential radiation risks that can indirectly contribute to the limited data availability in certain cases.

iii Poor resolution is also an issue in the medical imaging.

This makes use of models that have been trained on natural image data-sets like ImageNet. Much architecture like ResNet, EfficientNet and VGG can be used to optimize for medical applications and used as feature extractors. TL performs CNNs that are trained from scratch in medical imaging tasks with sample sizes. TL is an important feature where the source and data-sets are interrelated. It is possible to perform large, unrelated data-sets with moderately sized medical data. It is an important strategy for building accurate models along with limited labeled samples (21). Many recent studies highlight the use of ResNet and DenseNet to achieve strong generalization rather than other medical modalities due to their skip connections and effective usage. This choice depends mainly on the trade-off between accuracy and resources rather than clinical deployment (21, 22). While freezing older convolutional layers during training, it is less effective on small data-sets. However, there is still an issue with features obtained from natural photos not translating perfectly to medical domains, which calls for domain-specific pre-training on large medical corpora (23).

To overcome certain challenges, it is good to use layer-wise learning rate adaptation has been used in which early layers are updated with small learning rates, which helps to prevent general visual features, rather than depending on higher layers on medical data (24). In conclusion, studies show that TL consistently enhances performance on data-poor medical tasks. More comprehensive bench-marking of different designs and fine-tuning methods is needed to provide recommendations for small clinical data-sets.

Methodology

Data-sets

In this study, the selection of data sets was illustrated by three key criteria to guarantee reliability and clinical relevance: the accessibility of expert-annotated samples to promote DL training, a concentrate on chest x-ray images of COVID-19 and pneumonia owing to their importance in respiratory illness diagnosis, and the addition of balanced data sets with normal, pneumonia, and COVID-19 cases to diminish bias and improve model generalization. Data-sets are more than 300 medical image data-sets are in the collection. It seems to have challenges with the data between 2004 and 2020. The time of these medical image data-sets can be split into two. It started from 2013, providing the excellent AlexNet in the ILSVRC competition from 2012. The success of this data-set motivates more researchers to solve the major challenges with the different DL methods. In addition to these analyses, many researchers focus on analyzing medical images with the adoption of DL methods is another important driving factor that resonates for the release of another set of medical data-sets (25).

TABLE 1 | Summary of recent transfer learning (TL) approaches for medical image classification, including data sets, models, performance outcomes and limitations.

Study/reference	Year	Domain/modality	TL approach	Key insight/performance outcomes and contribution	Limitation/gap
Apostolopoulos and Mpesiana	2020	COVID-19 chest X-ray	ResNet50 pre-trained on ImageNet	Achieved >96% accuracy on small COVID data sets	Achieved >96% accuracy on small COVID data sets
Wang et al.	2021	Pneumonia X-ray	DenseNet121 TL	Improved pneumonia detection with >90% accuracy	Required significant computational power
Chowdhury et al.	2022	COVID-19 and pneumonia X-ray	VGG19 + ResNet TL	Highlighted benefit of ensemble TL models	Limited generalization to multi-institution data sets

TABLE 2 | Medical image datasets summary.

Category	Dataset	Modality	# Images/patches	Classes	Notes
Pneumonia	Kaggle chest X-ray (pediatric dataset)	X-ray	~5,863 images (train/test split used)	Normal, pneumonia	Standard benchmark dataset, publicly available
COVID-19	COVID-chest X-ray	X-ray	Hundreds	COVID-19	Annotated metadata, small but widely used
General chest	NIH chest X-ray 14	X-ray	100,000+	14 disease categories	Multi-label dataset

TABLE 3 | Major challenges for TL in medical imaging.

Challenge	Description	Potential solutions
Data scarcity	Limited annotated datasets in medical domain	Data augmentation, synthetic data generation
Domain shift	Medical images differ significantly from ImageNet	Domain adaptation, fine-tuning on medical-specific data
Interpretability	Black-box models reduce trust	Use Grad-CAM, SHAP, and other interpretability methods
Class imbalance	Normal vs. pneumonia samples may be skewed	Weighted loss functions, balanced sampling

Covid-19 chest X-ray data-sets

- COVID-ChestXray is a set of frontal and lateral chest X-ray images that include hundreds of COVID-19 cases and are annotated with meta-data (e.g., ICU status and survival).
- COVIDx data-set: A collection of multiple data-sets, including Italian Society of Medical and Interventional Radiology (SIRM), Radiological Society of North America (RSNA) pneumonia, Cohen's, and others, for multi-class classification (normal, pneumonia, and COVID-19). There are thousands of pictures in each class.
- The COVID-19 Radiography Database (Kaggle) is a balanced collection of images of COVID-19, viral pneumonia, and normal images that can be used for TL tasks (**Table 2**).

These data-sets are very useful for testing TL models on small amounts of COVID-19-related images.

Major challenges

The major challenges of TL are overparameterization means that deep CNNs like ResNet have improved accuracy, but they exhibit slow and inefficient training. Expensive computations need costly hardware to be implemented, but it gives a deeper architecture with excellent performance. Insufficient labeled data means it is hard to annotate images in comparison to natural images, which makes it difficult for models to generalize well. The approaches like lightweight architectures and multistage TL, can help to handle these issues (**Table 3**).

Research methodology: TL medical image classification

It is applied to gather memory dumps, images, and network packet captures. Medical imaging data-sets, such as MRI images, CT scans, are collected utilizing the freely accessible data repository. This model necessitates high-quality image data so that it will be beneficial to verify with labels. Data preprocessing is done in this for image re-sizing, normalization, etc., and it primarily ensures that information is ready for training. For maintaining data integrity, the hashing method is used. Data preservation is important in the TL model because it facilitates to store the reliability of original image, and copies can be produced for experimentation purposes. MD5, SHA256 are the hashing approaches that can be utilized to verify the data-set integrity and stop alterations. This stage guarantees that empirical results will be consistent and can be repeated additional through the study. For examination purpose, it utilizes forensic software like Volatility, WireShark, Autopsy. It also adopts some DL framework of Pytorch and TensorFlow. Data augmentation methods can be used so that images are flipped, rotated and scaled which facilitates to restrict generalization. Classification layers are updated so that it can able to use special medical imaging tasks. It involves evaluation and understanding of the images along with performance metrics such as accuracy, precision, area under curve (AUC), etc. The use of visual tools, such as receiver operating characteristic (ROC) curves, precision graphs, is generated to display the TL model's effectiveness. The results are documented in the well-structured reports, and they offer clear validation of how TL improves medical image classification (**Figure 2**).

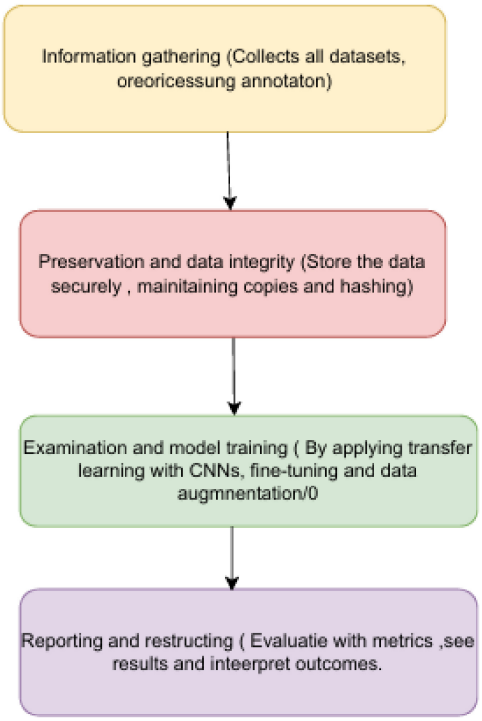

FIGURE 2 | TL medical image classification.

Forensic/data handling steps

It is applied to gather memory dumps, images, and network packet captures. Medical imaging data-sets, such as MRI images, CT scans, are collected utilizing the freely accessible data repository. This model necessitates high-quality image data so that it will be beneficial to verify with labels. Data preprocessing is done in this for image re-sizing, normalization, etc., and it primarily ensures that information is ready for training. For maintaining data integrity, the hashing method is used. Data preservation is important in the TL model because it facilitates to store the reliability of original image, and copies can be produced for experimentation purposes. MD5, SHA256 are the hashing approaches that can be utilized to verify the data-set integrity and stop alterations. This stage guarantees that empirical results will be consistent and can be repeated additional through the study. For examination purpose, it utilizes forensic software like Volatility, WireShark, Autopsy. It also adopts some DL framework of Pytorch and TensorFlow. Data augmentation methods can be used so that images are flipped, rotated and scaled which facilitates to restrict generalization. Classification layers are updated so that it can able to use special medical imaging tasks. It involves evaluation and understanding of the images along with performance metrics such as accuracy, precision, AUC, etc. The use of visual tools, such as ROC curves, precision graphs, is generated to display the TL model's effectiveness. The results are documented in the well-structured reports, and they offer clear validation of how TL improves medical image classification (**Table 4**).

Evaluation metrics

Evaluation metrics are quantitative measures used to test the performance of a classification model. Commonly applied in medical image analysis, these include: Accuracy, which indicated the rate of correctly classified samples; Precision, which quantifies how many of the predicted positives are essentially correct; Recall, which shows the capacity of the model to properly identify actual positives; F1-Score, the harmonic mean of accuracy and recall that balance both metrics; and AUC-ROC, which evaluates the models capability to distinguish between classes by plotting the true positive rate (TPR) against the false positive rate (FPR).

1. Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Precision (positive predictive value):

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

TABLE 4 | Forensic/data handling steps.

Step	Description	Tools/methods	Purpose
Information gathering	Collect memory dumps, magnetic resonance imaging (MRI), CT images	Freely accessible data repository.	Ensure high-quality labelled data
Preservation & chain of custody	Maintain dataset integrity	MD5, SHA256 hashing	Maintain data integrity and reproducibility
Examination	Data preprocessing, cleaning, augmentation	OpenCV, TensorFlow, Keras	Ensure clean and balanced dataset
Reporting and reconstruction	Summarize findings and model interpretability	Grad-cam visualizations, classification masks	Clinical usability and explainability

3. Recall (sensitivity/TPR):

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

4. F1-score (harmonic mean of precision and recall):

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

5. AUC-ROC:

The ROC cure plots:

TPR/recall:

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

FPR:

$$\text{FPR} = \frac{FP}{FP + TN}$$

The AUC is the area under the ROC curve:

$$\text{AUC} = \int_0^1 TPR\,(FPR)\,d(FPR)$$

Transfer learning for medical image classification

There is an experimental workflow that helps to estimate the DL pipeline for pneumonia classification using ResNet50 and TL, and a baseline CNN for additional comparison.

The workflow to classify pneumonia from chest X-ray images. The data-set was organized into test sets, validation, and training with data augmentation to enhance generalization.

Mount google drive to google colab

This step provides direct access to the files that are stored in the drive. It is beneficial for the processing tasks of images with large data-sets. It used to eliminate the need to repeatedly upload data to the Colab environment (**Appendices I** and **II**). It mainly allows any code in our notebook to access any files that are present in the drive (**Appendix III**). In this case, the data-sets are found in the /content/data/chest-x-ray folder. It provides the easy access to the data-set and enables the project to store, process and maintain it in all the phases (**Appendices IV** and **V**).

Importing the libraries

This step ensures that image processing involves the complex operations like filtering, image extraction, transformations, etc. It significantly reduces the complex development time and occurrence of potential errors. Importing several libraries not only helps in processing, but also effectively brings basic image editing to the advanced computer vision and reduces the heavy performance of machine learning tasks. In this model, importing all the necessary Python libraries to access experiments in DL, such as TensorFlow/Keras for neural networks, OpenCV for preprocessing images and NumPy for numerical computation and MatPlotLib for visualizations (**Appendix VI**).

Data-set preparation

This step is the important one because it needs much time to process cleaning, organizing and transforming the raw data into the specialized format, which is also suitable for the processing of machine learning algorithms. It ensures that handling of missing values, feature extraction, data formatting and finalize the data-set with high-quality feature, thus providing accurate and reliable models.

In this case, the chest X-ray pneumonia data-set is trained, validated and organized into folder. It ensures that processing aligns with the Keras image data generator and facilitates the training and unbiased testing process (**Appendix VII**).

Results

Data augmentation

This step is used to expand the trained data-set by generating new, altered versions of the existing images. It is particularly helpful in the real-world data collection, which is limited and

expensive. It enhances model robustness and generalization capabilities of machine learning, especially in DL models.

In this model, to minimize over-fitting, the data augmentation is used with the Keras ImageDataGenerator. This library is helpful to modify the images with zoom options, horizontal twists, flips, re-sizing, image scaling, etc. These can be applied only to the training data-set. The remaining test and validation data-sets are untouched and cannot able to alter the information (**Appendix VIII**).

Load data generators

The loading data generator allows us to randomly do image rotation without any specific degree between 0 and 360. It is an integer value with the rotation range argument. In this way, we can allow to feed data into Keras in real-time while training the model.

In this model, separator data generators are used for all three steps of data. Each generator specifies how to handle the tasks of images. It ensures efficient memory usage with a smooth transition flow.

Visualize the sample images

Visualizing sample images from an image processing data-set helps to understand the data characteristics. It includes various types of objects, lighting conditions and potential variations present. It helps to identify common image features and challenges within the data. It helps to reveal anomalies, inconsistencies within the images.

In this image processing model, random images from the training data-set are displayed. It helps to ensure that the model created has a quality image, and it will receive properly formatted inputs and prevent errors from occurring (**Table 5**).

Baseline CNN definition

CNNs are DL models that are designed to process data with a grid-like topology. It acts as the foundation of the most modern computer vision applications that are used to detect features within the visual data.

It is the kind of artificial neural network used for image recognition and processing because of its ability to recognize patterns in images.

In this step, a custom CNN was generated as the baseline model. It has multiple convolutional, pooling layers, thus help to extract spatial features. It is followed by a fully connected dense layer helps for classification. It mainly helps to reduce over-fitting and improve generalization (**Table 6**).

Compilation of baseline CNN

This is the image processing that serves as a simple standard model used to establish a performance benchmark for more complex thus enabling the evaluation of improvements and the quantification of their added value. The primary purpose provides the fundamental performance metric against which more advanced CNN architectures.

In this model, the baseline CNN is compiled using the Adam optimizer, performance metric, etc. This step enables the stable and efficient training for the pneumonia detection task.

Train baseline CNN

Train baseline CNN is helps to quickly converge the model during training and protect from inefficiency and enhance overall network stability (**Appendix IX**).

In this model, the baseline CNN was trained with five epochs using the validation data accuracy and information tracking after each epoch. It helps to identify the over-fitting early in the model processing.

Evaluate baseline CNN

This step is used to understand how the CNN generalizes the new unseen images, which helps to identify areas for improvement, and meets the specific performance goals for the given image processing application, such as classification or image recognition. Based on this evaluation, we can decide if the model needs more training, data augmentation to improve performance.

This step ensures that the baseline CNN model was evaluated on the unseen test data-sets. Performance accuracy was recorded, and it serves as the basis to measure further improvements based on the TL model (**Appendix X**).

Visualize baseline training curves

This is an important part of the image processing that serves to establish a performance basis to assess the complex models or the effectiveness of the model. These curves allow researchers to diagnose issues like over-fitting, under fitting, stability, convergence, and ultimately quantify improvements on the models.

In this model, graphs showing the training versus validation are plotted based on accuracy (**Figure 3**) and loss (**Figure 4**). These visualization gives deeper insights how into the model is learned and how it fits with the trained data (**Appendix XI**).

TABLE 5 | Data augmentation and preprocessing summary.

Step	Technique/method	Purpose	Notes
Data augmentation	Zoom, flips, rotation, scaling, resizing	Expand dataset, reduce overfitting	Applied only to training set
Data generators	ImageDataGenerator (Keras)	Efficient batch loading and augmentation	Reduces memory usage
Visualization	Matplotlib, Grad-CAM	Understand training and feature focus	Improves interpretability

TABLE 6 | Baseline CNN setup and training.

Aspect	Details	Purpose/notes
Model architecture	Custom CNN with convolutional layers, pooling layers, and fully connected dense layer	Extract spatial features and perform classification
Compilation	Optimizer: Adam, Loss: Binary CrossEntropy, Metrics: Accuracy	Standard setup for binary classification
Training	Epochs: 20, Batch size: 32	Used training + validation split
Evaluation	Accuracy: 84.29%	Good baseline model
Visualization	Training/validation curves, Grad-CAM	To analyze over fitting and interpretability

Load pretrained ResNet50

To load a pretrained ResNet50 model for image processing that we can utilize DL frameworks like Keras or PyTorch. It involves importing the necessary libraries, model instantiating and then preparing our input images for the model.

In this model, the architecture pertains to the ImageNet data-set with top classification layers removed. It allows features to be reused for medical image analysis (**Appendix XII**).

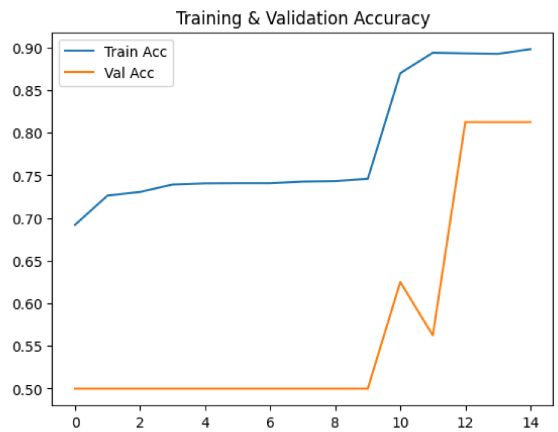

FIGURE 3 | Training and validation accuracy.

Freeze the convolutional base

Freezing the convolutional base in image processing, within the context of DL and CNNs, refers to the practice of preventing weights of convolutional layers from being updated during the training process. This technique is primarily employed in TL.

In this step, the CNN of ResNet50 was frozen so that it would not be updated during the training. It ensures that only the added layers will learn, and existing layer acts as the fixed feature extractor (**Appendix XIII**).

Add custom classification layers

Adding custom classification layers in the image processing in a DL model involves modifying the final layers of a pre-trained CNN. Compiling this model with an appropriate optimizer, such as Adam, loss function, and cross-entropy for multi-class.

In this model, custom dense layers were added to the top of the frozen ResNet50 base. This layer is used predominantly in classifying the images as Pneumonia, making the model task-specific rather than understanding the pre-trained visual features (**Appendix XIV**).

Compile transfer learning model

This step involves compiling a TL model in image processing by configuring the model for training after the architecture is defined and modified. Compiling is done by selecting an optimization algorithm to update the model's weights during training.

In this model, using the Adam optimizer, binary cross-entropy, and accuracy as the performance metric. This configuration not only acts as a baseline it also creates a fair and direct comparison with performance (**Appendix XIV**).

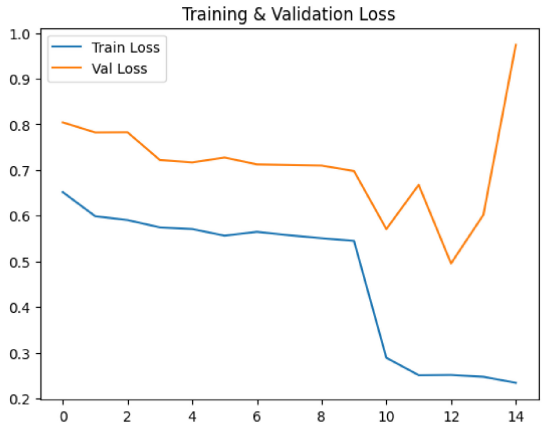

FIGURE 4 | Training and validation loss.

Train transfer learning model

The primary purpose of this step is to understand a pre-trained model with learned features for new, related work, thus leads to faster development, better performance with less data and improves quality and accuracy by utilizing the visual features of images. This model was trained for five epochs on the same training data-sets. It first understands the pre-trained knowledge and the ResNet59 converged faster and achieved accuracy in comparison with the baseline CNN.

Evaluate transfer learning model

This step is much needed in the TL model because it is used to improve performance, reduce training time and computation costs and overcome data scarcity by understanding the features such as semantic information, textures, etc. It accelerates the development with effective computer vision, image classification, medical imaging, etc. In this model, ResNet50 is fine-tuned and tested on the same data-set. Performance metrics were recorded, and it clearly shows improvement of ResNet50 is better in comparison with baseline CNN and confirms it is suitable for small data-sets.

Plot ResNet50 training curves

Plotting ResNet50 training curves illustrates the accuracy and loss curves during the training of the model using TensorBoard. In this step, training and validation curves for ResNet50 are plotted with accuracy and loss. It highlights the stable behavior with minimized over-fitting in comparison with the baseline CNN (**Figure 5**; **Appendix XV**).

Compare baseline vs. ResNet50

It means baseline model serves better in comparison with ResNet50. It is because baseline CNN provide foundational approach whereas ResNet50 use existing information to create accuracy in classifying image objects.

In this model, both models are evaluated simultaneously. Here, ResNet50 performs less in compare with baseline CNNs thus confirming better generalization, providing the effectiveness of using pre-trained architecture for classification (**Table 7**).

Grad-CAM implementation setup

Grad-CAM is the technique that is used in image processing to help us to visualize the regions of the image that the CNN focuses on to make specific prediction. It provides interpretability to DL models.

In this model, this technique is used to visualize the image-specific regions. To enhance, interpretability Grad-CAM was implemented.

Grad-CAM function definition

This function is the technique that is used to make CNN-based models more transparent, rather than visualizing the input regions, and it is very much used for predictions. It uses the specific gradient to create novel high-resolution and class-discriminative visualization. This function makes the black-box model more explainable. In this model, a custom function was created to compute heat maps from immediate convolutional activations and class specific gradients (**Appendix XVI**).

Generate grad-CAM for baseline CNN

This step involves visualizing the regions which has an input image that are helpful for specific class prediction. This Grad-Cam requires access to the feature maps and the gradients. It involves overlaying the resulting heat maps onto the original image. These heat maps revealed the baseline focused on detecting pneumonia (**Appendix XVII**).

Generating grad-CAM for ResNet50

This step involves Grad-CAM heatmap for this model. First it is important to train this model on an input image, and the last convolutional layer's output feature map. It results with the generation of a heatmap of important regions in an image (**Figure 6**; **Appendix XVIII**; **Table 8**).

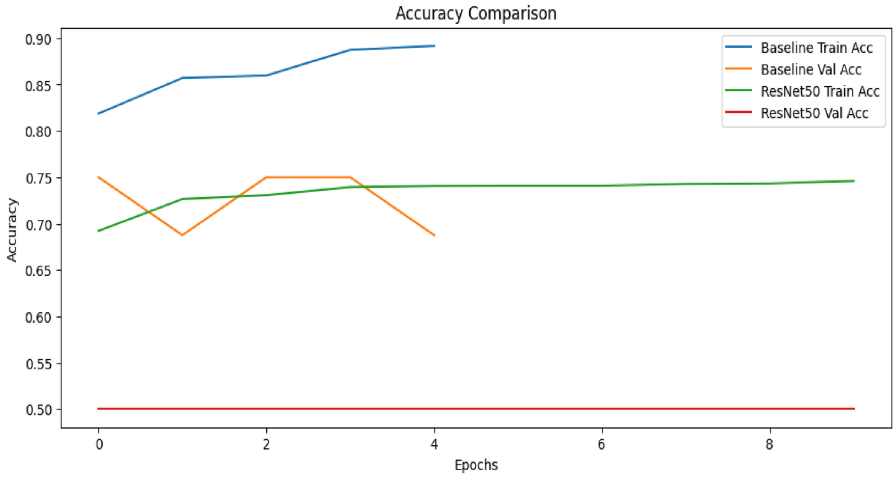

FIGURE 5 | Accuracy comparison.

Overlay heatmaps

This step involves with heatmap that helps to better visualize the volume of locations that are present within the data-set and helps in directing viewers towards the data visualization that matters most. These are superimposed onto the original X-ray images. It is used to visually represent the lung region.

Compare baseline vs. ResNet50 heatmaps

It involves with visualization of parts of the image that a model focuses on for classification (**Figure 7**; **Appendices XIX** and **XX**). In this comparison, ResNet50 exhibits deeper and more complex neural network features, whereas the baseline shows simpler and more generic visualization. ResNET50 localized pneumonia regions with accurate images and supports reliability (**Figures 8** and **9**).

TABLE 7 | Model performance comparison.

Model	Accuracy (%)	Notes
Baseline CNN	84.29%	Higher recall for Pneumonia (0.96), robust baseline
ResNet50 (TL)	86.22%	More balanced precision-recall, improved generalization

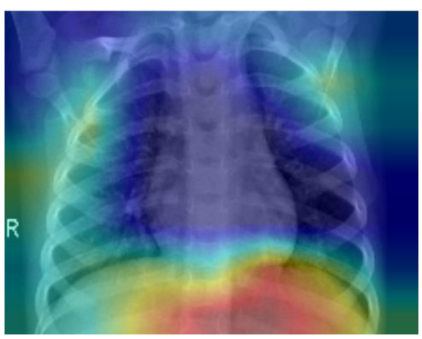

FIGURE 6 | Grad- CAM heatmap.

Classification Report - Baseline CNN

	precision	recall	f1-score	support
Normal	0.91	0.64	0.75	234
Pneumonia	0.82	0.96	0.88	390
accuracy			0.84	624
macro avg	0.87	0.80	0.82	624
weighted avg	0.85	0.84	0.84	624

FIGURE 7 | Classification report - baseline CNN.

Visualize random test sample

In this step, arbitrary test images are selected, and the Grad-Cam overlays heat maps are generated. This analysis illustrates robustness across patient's unseen scans (**Appendix XXI**; **Figure 10**).

Summarize model accuracy

In this model, baseline CNN and ResNet50 comparison, baseline attain accuracy 84% than ResNet50. It highlights the pre-trained deep architectures will exhibit more in compare with recent developing model (**Appendix XXII**).

Summarize results

From this analysis, we can ensure that ResNet50 works better as it tends to attain high precision with improved interoperability function. It gives a clear image visualization in the lung regions.

TABLE 8 | Grad-CAM visualization and interpretability.

Model	Grad-CAM applied	Observations/focus areas	Interpretation/notes
Baseline CNN	Yes	Heatmaps highlight pneumonia regions, but coarse	Useful for localization but less detailed
ResNet50	YES	Focuses sharply on affected lung regions	Provides finer granularity, more clinically relevant
Random test samples	Yes	Some false positives due to background noise	Highlights need for more robust training

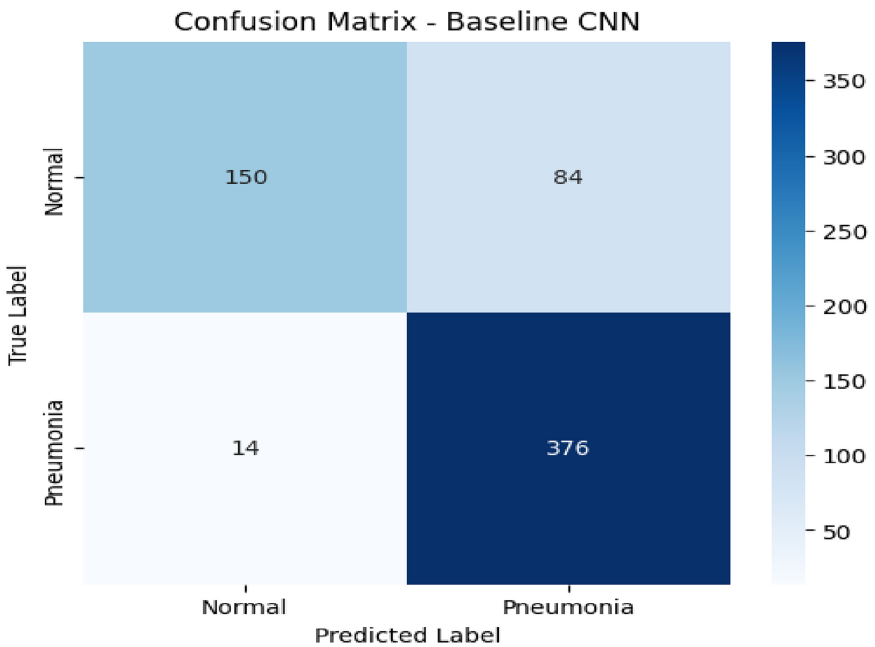

FIGURE 8 | Confusion matrix - baseline CNN.

Highlighting the clinical implications

ResNet50 has the most improved performance in comparison with all the methods. It suggests that this technique can be deployed in medical image screening for resource-limited settings (**Table 9**).

Discussion

The comparison between the baseline CNN and the ResNEt50 TL model underscores the purpose of using pre-trained architectures for medical image classification. The baseline CNN was 84.29% precise and is a valid citation point for comparison because it had a considerable recall (0.96) for pneumonia detection, which is clinically appropriate for scientific applications. Training CNNs from scratch with tiny medical datasets is problematic due to their shortcomings in generalization and their propensity to overfitting.

The data was fine-tines to achieve 86.22% accuracy, which illustrates a better tradeoff between precision and recall. Previous studies have revealed that TL is better than scratch-built models on tiny medical datasets, resulting in a COVID-19 detection precision of >96% using ResNet50. Also pointed out the usefulness of ensemble TL approaches. Although our results are Somewhat less accurate than these benchmarks, the advancement over the baseline CNN indicates the resilience of TL to limited and imbalanced datasets.

Interpretability through Grad-CAM revealed that ResNEt50 enhanced classification precision and pinpointed pathological regions with fine granularity than the baseline CNN. The capability to visualize strengthens trust in clinical adoption, where predictability is equally important as explainability.

This research advocates that TL is a practicable and effective way for locating medical images in environments with little data. To achieve high-quality performance similar to the highest outcomes in existing literature, it is recommended to improve model selection, dine-tune strategies, and domain adaptation.

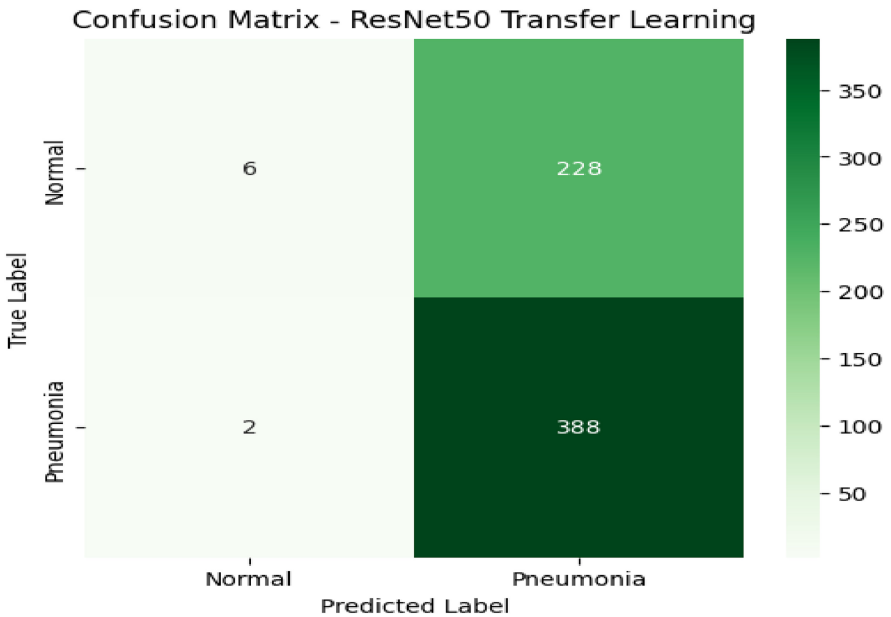

FIGURE 9 | Confusion matrix - ResNet50 TL.

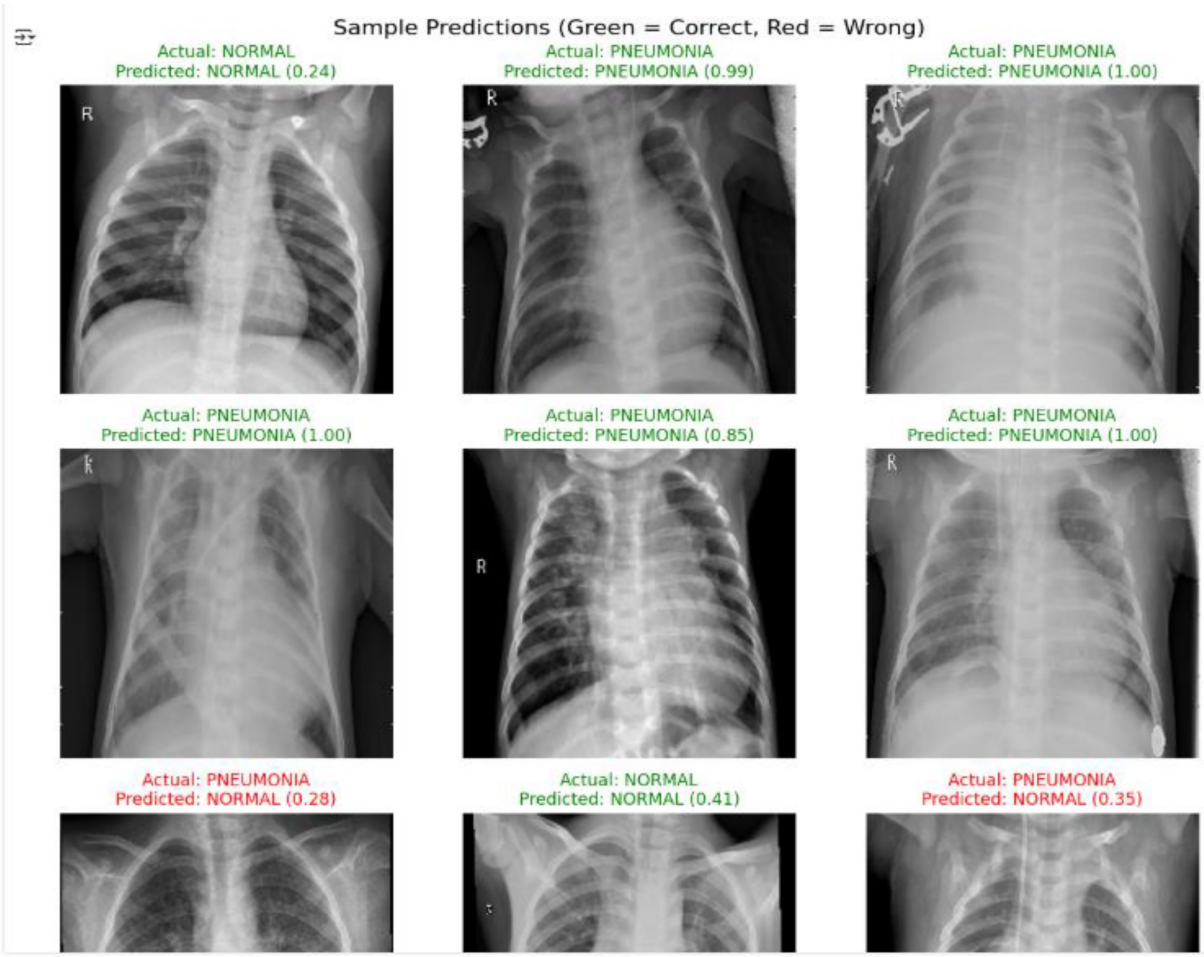

FIGURE 10 | Sample predictions.

TABLE 9 | Clinical relevance of model.

Model	Key advantage	Limitation	Clinical implication/use case
Baseline CNN	Simple, interpretable, lightweight	Lower generalization, may miss subtle cases	Educational tool, initial screening
ResNet50 (TL)	Higher accuracy, robust detection	Computationally heavy, risk of over fitting on small data	Useful for decision support in radiology

Conclusion

This research explored the purpose of TL to categorize medical images using COVID-19 chest X-ray datasets, comparing a baseline CNN to ResNet50. The baseline CNN was able to achieve 84.29% accuracy, but ResNet50's performance improved to 86.22%, concluding that TL is effective in optimizing classification accuracy, robustness, and interpretability with limited annotated datasets. When compared with prior studies, reported >96% accuracy for COVID-19 detection with ResNet50, and achieved >90% accuracy using DenseNet121, this findings illustrate competitive performance, though nit yet surpassing the highest benchmarks. This research corroborates that TL invariability improves generalization over baseline CNN's even in instances with limited resources. In summary, TL not just bridges the gap synthesized by limited annotated medical datasets but also presents a scalable, accurate, and explicable means to advance AI-assisted healthcare solutions.

Funding

The authors declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Mehmood A, Yang S, Feng Z, Wang M, Ahmad AS, Khan R, et al. A transfer learning approach for early diagnosis of Alzheimer's disease on MRI images. *Neuroscience.* (2021) 460:43–52.
2. Malik H, Farooq MS, Khelifi A, Abid A, Qureshi JN, Hussain M. A comparison of transfer learning performance versus health experts in disease diagnosis from medical imaging. *IEEE Access.* (2020) 8:139367–86.
3. Tanveer M, Rashid AH, Ganaie MA, Reza M, Razzak I, Hua KL. Classification of Alzheimer's disease using ensemble of deep neural networks trained through transfer learning. *IEEE J Biomed Health Informat.* (2021) 26(4):1453–63.
4. Alzubaidi L, Al-Amidie M, Al-Asadi A, Humaidi AJ, Al-Shamma O, Fadhel MA, et al. Novel transfer learning approach for medical imaging with limited labeled data. *Cancers.* (2021) 13(7):1590.
5. Kim HE, Cosa-Linan A, Santhanam N, Jannesari M, Maros ME, Ganslandt T. Transfer learning for medical image classification: a literature review. *BMC Med Imaging.* (2022) 22(1):69.
6. Matsoukas C, Haslum JF, Sorkhei M, Söderberg M, Smith K. What makes transfer learning work for medical images: feature reuse & other factors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (2022). p. 9225–34.
7. Hosna A, Merry E, Gyalmo J, Alom Z, Aung Z, Azim MA. Transfer learning: a friendly introduction. *J Big Data.* (2022) 9(1):102.
8. Mehmood S, Ghazal TM, Khan MA, Zubair M, Naseem MT, Faiz T, et al. Malignancy detection in lung and colon histopathology images using transfer learning with class selective image processing. *IEEE Access.* (2022) 10:25657–68.
9. Taresh MM, Zhu N, Ali TAA, Hameed AS, Mutar ML. Transfer learning to detect COVID-19 automatically from X-Ray images using convolutional neural networks. *Int J Biomed Imaging.* (2021) 2021(1):8828404.
10. Apostolopoulos ID, Mpesiana TA. Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Phys Eng Sci Med.* (2020) 43(2):635–40.
11. Valverde JM, Imani V, Abdollahzadeh A, De Feo R, Prakash M, Ciszek R, et al. Transfer learning in magnetic resonance brain imaging: a systematic review. *J Imaging.* (2021) 7(4):66.
12. Alzubaidi L, Fadhel MA, Al-Shamma O, Zhang J, Santamaría J, Duan Y, et al. Towards a better understanding of transfer learning for medical imaging: a case study. *Appl Sci.* (2020) 10(13):4523.
13. Horry MJ, Chakraborty S, Paul M, Ulhaq A, Pradhan B, Saha M, et al. COVID-19 detection through transfer learning using multimodal imaging data. *IEEE Access.* (2020) 8:149808–24.
14. Pathak Y, Shukla PK, Tiwari A, Stalin S, Singh S. Deep transfer learning based classification model for COVID-19 disease. *IRBM.* (2022) 43(2):87–92.
15. Bansal M, Kumar M, Sachdeva M, Mittal A. Transfer learning for image classification using VGG19: caltech-101 image data set. *J Ambient Intel Hum Comput.* (2023) 14(4):3609–20.
16. Liang G, Zheng L. A transfer learning method with deep residual network for pediatric pneumonia diagnosis. *Comput Methods Prog Biomed.* (2020) 187:104964.
17. Salehi AW, Khan S, Gupta G, Alabduallah BI, Almjally A, Alsolai H, et al. (2023). A study of CNN and transfer learning in medical imaging: advantages, challenges, future scope. *Sustainability.* (2020) 15(7):5930.
18. Liu X, Song L, Liu S, Zhang Y. A review of deep-learning-based medical image segmentation methods. *Sustainability.* (2021) 13(3):1224.
19. Hossain MB, Iqbal SHS, Islam MM, Akhtar MN, Sarker IH. Transfer learning with fine-tuned deep CNN ResNet50 model for classifying

COVID-19 from chest X-ray images. *Informat Med Unlocked.* (2022) 30:100916.

20. Wang J, Zhu H, Wang SH, Zhang YD. A review of deep learning on medical image analysis. *Mobile Netw Appl.* (2021) 26(1):351–80.
21. Abbas A, Abdelsamea MM, Gaber MM. Detrac: transfer learning of class decomposed medical images in convolutional neural networks. *IEEE Access.* (2020) 8:74901–13.
22. Srinivas C, Nandini Prasad KS, Zakariah M, Alothaibi YA, Shaukat K, Partibane B, et al. Deep transfer learning approaches in performance analysis of brain tumor classification using MRI images. *J Health Care Eng.* (2022) 2022(1):3264367.
23. Gour N, Khanna P. Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network. *Biomed Sig Process Control.* (2021) 66:10 2329.
24. Kaur T, Gandhi TK. Deep convolutional neural networks with transfer learning for automated brain image classification. *Mach Vision Appl.* (2020) 31(3):20.
25. Mei X, Liu Z, Robson PM, Marinelli B, Huang M, Doshi A, et al. RadImageNet: an open radiologic deep learning research data-set for effective transfer learning. *Radiol Artific Intel.* (2022) 4(5):e21 0315.

Appendix

I. .STEP 1:

```
# Step 1: Install Kaggle API (Colab usually doesn't have it pre-installed)
!pip install -q kaggle
```

II. .STEP 2:

```
# Step 2: Upload kaggle.json (API key file)
from google.colab import files
files.upload()  # <— Upload kaggle.json from your computer
```

III. .STEP 3:

```
# Step 3: Move kaggle.json to the correct directory
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

IV. .STEP 4:

```
# Step 4: Download dataset from Kaggle
!kaggle datasets download -d paultimothymooney/chest-xray-pneumonia -p ./data
```

V. .STEP 5:

```
# Step 5: Unzip dataset
import zipfile
with zipfile.ZipFile("./data/chest-xray-pneumonia.zip", 'r') as zip_ref:
    zip_ref.extractall("./data")
print("Dataset downloaded and extracted successfully!")
```

VI. .STEP 6:

```
# Step 6: Import libraries
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

VII. .STEP 7:

```
# Step 7: Define paths
train_dir = "./data/chest_xray/train"
val_dir = "./data/chest_xray/val"
test_dir = "./data/chest_xray/test"
```

VIII. .STEP 8:

```
# Step 8: Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./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
)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        train_dir, target_size=(224,224), batch_size=32, class_mode='binary'
)
val_generator = val_datagen.flow_from_directory(
        val_dir, target_size=(224,224), batch_size=32, class_mode='binary'
)
test_generator = test_datagen.flow_from_directory(
        test_dir, target_size=(224,224), batch_size=32, class_mode='binary', shuffle=False
)
```

IX. .STEP 9:

Step 9: Train a Baseline CNN (No Transfer Learning)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Baseline CNN Model (without transfer learning)
baseline_model = Sequential([
                Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

baseline_model.compile(optimizer='adam',
            loss='binary_crossentropy',
            metrics=['accuracy'])

print(baseline_model.summary())

# Train baseline CNN
history_baseline = baseline_model.fit(
    train_generator,
    epochs=5,
```

```
    validation_data=val_generator
)
# Evaluate baseline CNN
loss, acc = baseline_model.evaluate(test_generator)
print(f"Baseline CNN Test Accuracy: {acc*100:.2f}%")
```

X. .STEP 10:

```
#  Step 10: Evaluate on Test Data
loss, acc = model.evaluate(test_generator)
print(f"Test Accuracy: {acc*100:.2f}%")
```

XI. .STEP 11:

```
# Step 11: Plot Accuracy & Loss
plt.plot(history.history['accuracy'] +
history_fine.history['accuracy'], label="Train Acc")
plt.plot(history.history['val_accuracy'] +
history_fine.history['val_accuracy'], label="Val Acc")
plt.legend()
plt.title("Training & Validation Accuracy")
plt.show()

plt.plot(history.history['loss'] + history_fine.history['loss'],
label="Train Loss")
plt.plot(history.history['val_loss'] +
history_fine.history['val_loss'], label="Val Loss")
plt.legend()
plt.title("Training & Validation Loss")
plt.show()
```

XII. .STEP 12:

```
# Step 12: Train Transfer Learning Model (ResNet50)

from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten,
Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

# Load pre-trained ResNet50 model (without top classifier
layers)
base_model = ResNet50(weights="imagenet",
include_top=False, input_shape=(224,224,3))

# Freeze base model layers (so only custom layers train
initially)
for layer in base_model.layers:
layer.trainable = False

# Add custom layers on top
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation="relu")(x)
x = Dropout(0.5)(x)
predictions = Dense(1, activation="sigmoid")(x)

model = Model(inputs=base_model.input,
outputs=predictions)

# Compile
model.compile(optimizer=Adam(learning_rate=0.0001),
        loss="binary_crossentropy",
        metrics=["accuracy"])
# Train the transfer learning model
history_resnet = model.fit(
  train_generator,
  validation_data=val_generator,
  epochs=5
)
# Evaluate on test data
loss_resnet, acc_resnet = model.evaluate(test_generator)
print(f"ResNet50 Transfer Learning Test Accuracy:
{acc_resnet*100:.2f}%")
```

XIII STEP 13:

```
# Step 13: Build Model (Transfer Learning with ResNet50)
base_model = ResNet50(weights="imagenet",
include_top=False, input_shape=(224,224,3))
base_model.trainable = False  # Freeze base model layers

model = Sequential([
  base_model,
  GlobalAveragePooling2D(),
  Dropout(0.5),
  Dense(128, activation="relu"),
  Dropout(0.3),
  Dense(1, activation="sigmoid")
])
model.compile(optimizer=Adam(learning_rate=0.0001),
        loss="binary_crossentropy",
        metrics=["accuracy"])
model.summary()
```

XIV STEP 14:

```
# Step 14: Fine-tuning (Unfreeze last few ResNet50 layers
for better accuracy)
base_model.trainable = True
for layer in base_model.layers[:-30]:  # keep most frozen
  layer.trainable = False

model.compile(optimizer=Adam(learning_rate=1e-5),
        loss="binary_crossentropy",
        metrics=["accuracy"])

history_fine = model.fit(
  train_generator,
  validation_data=val_generator,
  epochs=5
)
```

XV STEP 15:

```
#Step 15: Plot Training Curves

plt.figure(figsize=(12,5))

# Baseline CNN
plt.plot(history_baseline.history['accuracy'],
label="Baseline Train Acc")
```

```
plt.plot(history_baseline.history['val_accuracy'], label="Baseline Val Acc")

# ResNet50
plt.plot(history.history['accuracy'], label="ResNet50 Train Acc")
plt.plot(history.history['val_accuracy'], label="ResNet50 Val Acc")

plt.title("Accuracy Comparison")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

XVI STEP 16:

```
# Step 16: Grad-CAM Function
import cv2
from tensorflow.keras.models import Model
def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):
    # Create model mapping input -> last conv layer outputs & predictions
    grad_model = Model(
        [model.inputs],
        [model.get_layer(last_conv_layer_name).output, model.output]
    )
    # Forward pass
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(predictions[0])
        class_channel = predictions[:, pred_index]
    # Compute gradients
    grads = tape.gradient(class_channel, conv_outputs)
    # Mean intensity of gradients across each feature map
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
    # Weight activation maps with pooled gradients
    conv_outputs = conv_outputs[0]
    heatmap = conv_outputs @ pooled_grads[..., tf.newaxis]
    heatmap = tf.squeeze(heatmap)
    # Normalize between 0 and 1
    heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
    return heatmap.numpy()
def display_gradcam(img_path, model, last_conv_layer_name="conv5_block3_out"):
    # Load and preprocess image
    img = tf.keras.utils.load_img(img_path, target_size=(224, 224))
    img_array = tf.keras.utils.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0
    # Generate heatmap
    heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)
    # Overlay heatmap on image
    img = cv2.imread(img_path)
    img = cv2.resize(img, (224, 224))
    heatmap = cv2.resize(heatmap, (img.shape[1], img.shape[0]))
    heatmap = np.uint8(255 * heatmap)
    heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)
    superimposed_img = cv2.addWeighted(img, 0.6, heatmap, 0.4, 0)
    plt.figure(figsize=(6,6))
    plt.imshow(cv2.cvtColor(superimposed_img, cv2.COLOR_BGR2RGB))
    plt.axis("off")
    plt.show()
```

XVII STEP 17:

```
# Step 17: Grad-CAM on Baseline CNN
import numpy as np
from tensorflow.keras.preprocessing import imag
# Make sure the model has seen an input before Grad-CAM
_ = baseline_model.predict(np.zeros((1,224,224,3)))
# Pick test image
test_dir = "/content/data/chest_xray/test/PNEUMONIA"
sample_img = random.choice(os.listdir(test_dir))
img_path = os.path.join(test_dir, sample_img)
print("Baseline CNN - Grad-CAM for:", sample_img)
# Run Grad-CAM
display_gradcam(img_path, baseline_model, last_conv_layer_name="conv2d_2")
```

XVIII STEP 18:

```
# Step 18: Grad-CAM on ResNet50 Transfer Learning
test_dir = "/content/data/chest_xray/test/NORMAL"
sample_img = random.choice(os.listdir(test_dir))
img_path = os.path.join(test_dir, sample_img)
print("ResNet50 - Grad-CAM for:", sample_img)
display_gradcam(img_path, model, last_conv_layer_name="conv5_block3_out")
```

XIX STEP 19:

```
# Step 19: Confusion Matrix & Classification Report
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
# Get predictions
y_pred = model.predict(test_generator)
y_pred_classes = (y_pred > 0.5).astype("int32").reshape(-1)  # Convert probabilities → class labels
y_true = test_generator.classes
# Confusion Matrix
cm = confusion_matrix(y_true, y_pred_classes)
```

```
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
        xticklabels=["Normal", "Pneumonia"],
        yticklabels=["Normal", "Pneumonia"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
# Classification Report
print("? Classification Report:\n")
print(classification_report(y_true, y_pred_classes, target_names=["Normal", "Pneumonia"]))
```

XX STEP 20:

```
# Step 20: Confusion Matrix (ResNet50 Transfer Learning)
cm_resnet = confusion_matrix(y_true, y_pred_resnet)
plt.figure(figsize=(6,5))
sns.heatmap(cm_resnet, annot=True, fmt='d', cmap='Greens',
        xticklabels=['Normal', 'Pneumonia'],
        yticklabels=['Normal', 'Pneumonia'])
plt.title("Confusion Matrix - ResNet50 Transfer Learning")
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
print("Classification Report - ResNet50 Transfer Learning")
print(classification_report(y_true, y_pred_resnet, target_names=['Normal', 'Pneumonia']))
```

XXI STEP 21:

```
# Step 21: Sample Predictions Visualization
import random
import os
# Get filenames from test set
filenames = test_generator.filenames
true_labels = test_generator.classes
class_labels = list(test_generator.class_indices.keys())
# Pick 9 random samples
plt.figure(figsize=(12, 12))
for i in range(9):
    idx = random.randint(0, len(filenames) - 1)
  img_path = os.path.join(test_dir, filenames[idx])
    # Load and preprocess the image (resize to 224x224 for ResNet50)
        img = tf.keras.utils.load_img(img_path, target_size=(224, 224))
    img_array = tf.keras.utils.img_to_array(img) / 255.0
  img_array = np.expand_dims(img_array, axis=0)
    # Prediction
    pred_prob = model.predict(img_array, verbose=0)[0][0]
  pred_class = 1 if pred_prob > 0.5 else 0
    # Plot
    plt.subplot(3, 3, i + 1)
        plt.imshow(tf.keras.utils.load_img(img_path, target_size=(224, 224)))
    plt.axis("off")
    plt.title(
      f"Actual: {class_labels[true_labels[idx]]}\nPredicted: {class_labels[pred_class]} ({pred_prob:.2f})",
        color="green" if pred_class == true_labels[idx] else "red"
  )
  plt.suptitle("Sample Predictions (Green = Correct, Red = Wrong)", fontsize=16)
  plt.tight_layout()
  plt.show()
```

XXII STEP 22:

```
# Step 22: Compare Both Models(Compare Baseline CNN vs ResNet50)
loss_base, acc_base = baseline_model.evaluate(test_generator verbose=0)
loss_resnet, acc_resnet = model.evaluate(test_generator, verbose=0)
print("?? Model Performance Comparison:")
print(f"Baseline CNN Accuracy: {acc_base*100:.2f}%")
print(f"ResNet50 Transfer Learning Accuracy: {acc_resnet*100:.2f}%")
```