PNEUMONIA DETECTION USING DEEP LEARNING ON CHEST X-RAY IMAGES

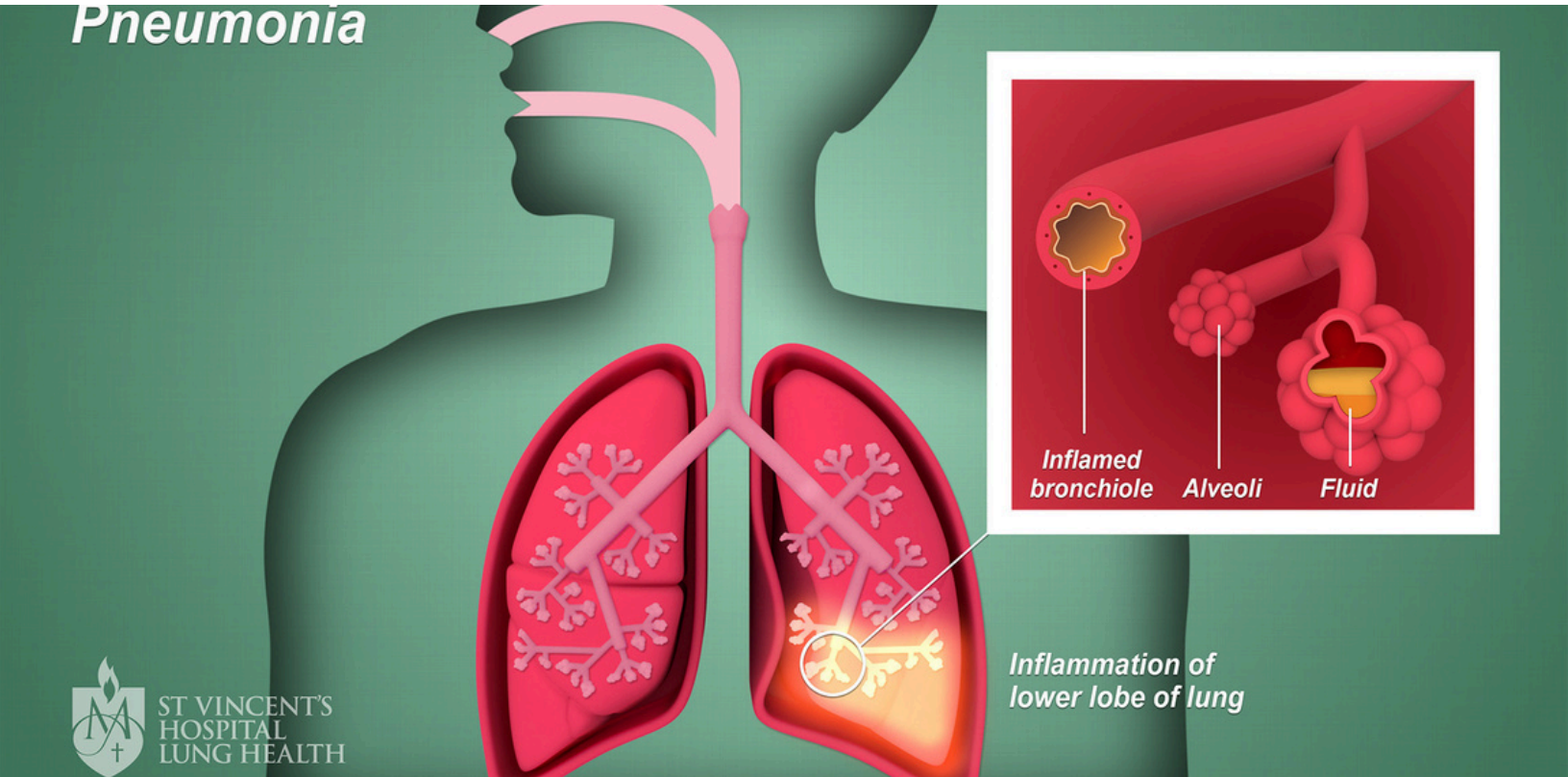
GROUP MEMBERS: Arshpreet Singh (C0908137), Daljeet Kaur (C0892746), Mohammed Moin (C0893375), Yash Sahu (C0895287), Parth Lanukia (C0895943)

TABLE OF CONTENT

INTRODUCTION

Pneumonia is a severe respiratory infection that affects the lungs, causing inflammation and fluid accumulation in the alveoli. Early detection is critical for effective treatment, but the symptoms can be similar to other respiratory conditions, making diagnosis challenging. Chest X-rays are commonly used for diagnosis, but interpreting them requires expertise and can be time-consuming.

This project aims to leverage deep learning techniques, particularly Convolutional Neural Networks (CNNs), to automate the detection of pneumonia from chest X-ray images. CNNs have proven effective in image classification tasks due to their ability to capture spatial hierarchies in images through layers of convolutions, making them suitable for medical imaging applications. The goal is to develop a model that can classify X-ray images as either 'Normal' or 'Pneumonia' with high accuracy, potentially assisting healthcare professionals in making quicker and more accurate diagnoses.



DATA PREPARATION

Dataset Overview

The dataset used in this project is a publicly available collection of chest X-ray images, categorized into two classes: 'Normal' and 'Pneumonia'. The dataset includes images of patients with bacterial and viral pneumonia, as well as healthy individuals. The dataset is divided into three main parts:

- Training Set: Used to train the model. It contains the majority of the data, allowing the model to learn the features that distinguish normal lungs from those affected by pneumonia.
- Validation Set: Used to fine-tune the model during training. It helps in assessing the model's performance on unseen data and prevents overfitting by providing a check on how the model is generalizing.
- Test Set: Used to evaluate the final model performance. This set is kept completely separate from the training process to provide an unbiased evaluation of the model's accuracy.

The images vary in size and quality, and preprocessing is required to standardize them for input into the CNN.

Dataset Structure

The dataset is organized into directories as follows:

- Train Folder: Contains subfolders NORMAL and PNEUMONIA, each with the corresponding X-ray images.
- Validation Folder: Similar structure to the training folder, but with fewer images, used for model validation during training.
- Test Folder: Contains the final set of images used to evaluate the model.

This organization allows for easy access and loading of images during the model development process.

DATA PREPARATION

Data Loading and Preprocessing

Data loading is done using TensorFlow's ImageDataGenerator, which allows for real-time data augmentation during training. The following preprocessing steps are applied:

- Resizing: All images are resized to a consistent shape of 64x64 pixels to match the input shape required by the CNN model.
- Normalization: Pixel values are scaled to a range of [0, 1] by dividing by 255. This helps in faster convergence during training by standardizing the input data.
- Augmentation: Techniques such as rotation, zoom, and horizontal flipping are applied to artificially increase the size of the training set. This helps the model generalize better by learning to recognize pneumonia from images with slight variations.

DATA VISUALIZATION

Sample Images from the Dataset

Visualizing the data is a crucial step to understand the variability within the dataset and to confirm that the images are correctly labeled. In this project, random samples from both the 'Normal' and 'Pneumonia' classes are visualized. This visualization serves multiple purposes:

- Understanding the Data: By visualizing the images, we can get an intuitive understanding of the differences between normal and infected lungs, which helps in designing the model architecture.
- Data Quality Check: It ensures that the images are correctly labeled and that there are no obvious anomalies in the dataset.

Comparison between Normal and Pneumonia Cases

Two sample images are selected, one from the 'Normal' class and the other from the 'Pneumonia' class.

- Normal X-ray: Typically, normal X-ray images show clear lung fields with no obstructions or cloudiness. The heart and diaphragm are clearly visible, and there is no evidence of fluid or consolidation in the lungs.
- Pneumonia X-ray: In contrast, X-ray images showing pneumonia often have visible areas of opacity, indicating fluid or consolidation in the lungs. These areas appear as white spots or patches and can vary in size and location depending on the severity of the infection.

This visual comparison highlights the challenge in detecting pneumonia, as the differences can be subtle, requiring the CNN to capture fine details in the images.

MODEL BUILDING

Convolutional Neural Network (CNN) Architecture

The CNN architecture developed for this project is a multi-layered model designed to automatically learn the hierarchical features in the X-ray images. The architecture is composed of the following key layers:

- Convolutional Layers: The first few layers of the CNN are convolutional layers that apply filters to the input images to extract low-level features such as edges and textures. These layers are crucial for capturing the spatial features in the X-ray images.
  - Conv2D (32 filters, 3x3 kernel, ReLU activation): The first convolutional layer extracts 32 feature maps from the input image using a 3x3 filter. The ReLU (Rectified Linear Unit) activation function introduces non-linearity, enabling the network to learn more complex patterns.
  - Conv2D (32 filters, 3x3 kernel, ReLU activation): A second convolutional layer further refines the features extracted by the first layer, improving the model's ability to recognize patterns associated with pneumonia.
- MaxPooling Layers: Following each convolutional layer, a MaxPooling layer is used to downsample the feature maps, reducing the dimensionality and the computational load.
  - MaxPooling2D (2x2 pool size): This layer reduces the size of the feature maps by taking the maximum value from 2x2 patches. This operation helps in extracting the most important features and reduces the risk of overfitting.
- Flatten Layer: The output from the final convolutional layer is flattened into a single vector, which is then fed into the fully connected layers for classification.
- Fully Connected Layers: These layers, also known as dense layers, perform the final classification.
  - Dense (128 units, ReLU activation): This fully connected layer takes the flattened feature map and processes it further with 128 neurons. The ReLU activation ensures that the model can capture complex patterns.
  - Dense (1 unit, Sigmoid activation): The output layer consists of a single neuron with a sigmoid activation function, which outputs a probability score indicating the likelihood of pneumonia.

Layer-wise Explanation

Each layer in the CNN plays a specific role:

- Convolutional Layers: Responsible for feature extraction. The filters learn to detect edges, textures, and more complex patterns as the depth of the network increases.
- MaxPooling Layers: Used for downsampling, reducing the computational complexity and helping the model to become invariant to small translations in the image.
- Flatten Layer: Converts the 2D feature maps into a 1D vector, preparing it for the fully connected layers.
- Fully Connected Layers: Perform the final classification based on the features extracted by the convolutional layers.

TRAINING THE MODEL

Model Compilation

Before training the model, it needs to be compiled, which involves selecting the optimizer, loss function, and evaluation metrics. For this project:

- Optimizer: The Adam optimizer is chosen for its efficiency and ability to handle sparse gradients. It combines the benefits of both the AdaGrad and RMSProp algorithms, making it well-suited for this task.
- Loss Function: Binary cross-entropy is used as the loss function because this is a binary classification problem. The loss function measures the difference between the predicted probabilities and the actual labels, guiding the model during training to minimize this difference.
- Metrics: Accuracy is used as the primary metric to evaluate the model's performance. This metric measures the proportion of correctly classified images out of the total number of images.

Training Procedure

The model is trained using the training data over a fixed number of epochs, where an epoch is a complete pass through the entire training dataset. The training process involves:

- Epochs: The number of complete passes through the training data. More epochs generally lead to better performance, but the model may overfit if trained for too long.
- Batch Size: The number of training samples used in one forward/backward pass. A smaller batch size leads to more updates per epoch, which can improve learning but also increases training time.
- Validation: After each epoch, the model is evaluated on the validation set. The validation accuracy and loss are monitored to detect overfitting. If the validation accuracy decreases while the training accuracy increases, it indicates that the model is starting to memorize the training data rather than generalizing from it.

Optimizer and Loss Function

- Adam Optimizer: Adam combines the advantages of two other popular optimization methods: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). It computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.
- Binary Cross-Entropy: This loss function is specifically designed for binary classification problems. It calculates the loss for each class label (0 or 1) by comparing the predicted probability with the actual class label.

MODEL EVALUATION

Performance Metrics

The model's performance is evaluated using several metrics:

- Accuracy: The ratio of correctly predicted instances to the total instances. Accuracy is a straightforward measure but may not always be sufficient, especially in cases where the classes are imbalanced.
- Precision: The ratio of true positive predictions to the total predicted positives. Precision is crucial when the cost of a false positive is high.
- Recall (Sensitivity): The ratio of true positive predictions to the total actual positives. Recall is critical in medical diagnosis, where missing a positive case (false negative) could have serious consequences.
- F1-Score: The harmonic mean of precision and recall, providing a single metric that balances both concerns.

Accuracy and Loss Curves

During training, the accuracy and loss for both the training and validation sets are plotted to monitor the model's progress. The following observations are typical:

- Training Accuracy and Loss: Typically increase and decrease, respectively, over time as the model learns the features in the training data.
- Validation Accuracy and Loss: Provide a check on the model's ability to generalize to unseen data. If the validation accuracy plateaus or begins to decrease while training accuracy continues to increase, it may indicate overfitting.

Confusion Matrix

A confusion matrix is a table used to describe the performance of a classification model. It includes:

- True Positives (TP): Cases where the model correctly predicts pneumonia.
- True Negatives (TN): Cases where the model correctly predicts the absence of pneumonia.
- False Positives (FP): Cases where the model incorrectly predicts pneumonia.
- False Negatives (FN): Cases where the model incorrectly predicts the absence of pneumonia.

The confusion matrix is crucial for understanding where the model is making errors, particularly in distinguishing between false positives and false negatives, which have different implications in medical diagnosis.

HYPERPARAMETER TUNING

Tuning Strategies

Hyperparameter tuning involves systematically adjusting the model's parameters to improve performance. Common strategies include:

- Grid Search: Testing a range of values for each hyperparameter in a structured manner. Although exhaustive, this method can be computationally expensive.
- Random Search: A more efficient approach that randomly samples hyperparameter values. It often finds good configurations with less computational cost than grid search.
- Bayesian Optimization: A probabilistic model is built for the objective function, and hyperparameter values are selected to maximize this function. This method is more sophisticated and can find optimal settings with fewer evaluations.

Hyperparameters tuned in this project include:

- Learning Rate: Adjusting the learning rate to control the speed at which the model learns.
- Batch Size: Tuning the batch size to balance between training time and model performance.
- Number of Filters: Changing the number of filters in the convolutional layers to affect the complexity of the feature maps.

Impact on Model Performance

The tuning process showed that certain configurations significantly improved the model's performance:

- Learning Rate: A lower learning rate led to more stable convergence, reducing the risk of overshooting the minima.
- Batch Size: A batch size of 32 was found to provide a good balance between training time and model accuracy.
- Number of Filters: Increasing the number of filters in the convolutional layers enhanced the model's ability to capture intricate details in the X-ray images, leading to better classification accuracy.

MODEL DEPLOYMENT WITH STREAMLIT

Introduction to Streamlit

Streamlit is an open-source app framework designed specifically for creating and sharing data science applications. It allows developers to turn Python scripts into interactive web applications easily and is particularly suited for deploying machine learning models. In this project, Streamlit was used to deploy the pneumonia detection model, making it accessible through a user-friendly web interface.

Streamlit Application Structure

The Streamlit application was designed to:

- Upload X-ray Images: Users can upload chest X-ray images through the app interface.
- Model Prediction: The app processes the uploaded image and uses the trained CNN model to predict whether the image shows signs of pneumonia.
- Display Results: The prediction results are displayed on the interface, along with the probability score indicating the likelihood of pneumonia.

Key Features of the Streamlit App

- Interactive Interface: Users can interact with the model by uploading different X-ray images and receiving immediate feedback.
- Real-time Prediction: The model predicts pneumonia presence in real-time as soon as the image is uploaded.
- Visualization: The app can also display the input image along with the predicted label, enhancing user understanding of the results.

Deployment Process

1. Model Export: The trained CNN model was saved in a compatible format (.keras file) using TensorFlow/Keras.
2. Streamlit Script: A Python script was written to load the model, preprocess uploaded images, and handle user inputs through Streamlit.
3. Hosting: The Streamlit app was hosted on a server, making it accessible via a web browser. This allowed users to interact with the model without needing to install any dependencies on their local machines.

CONCLUSION

Summary of Results

The CNN model developed in this project successfully classifies chest X-ray images as 'Normal' or 'Pneumonia' with high accuracy. The confusion matrix analysis showed that the model makes few false positives and false negatives, indicating strong generalization capabilities. The accuracy and loss curves suggest that the model is well-trained, with minimal overfitting observed.
Potential Improvements and Future Work
While the model performs well, there are several areas for potential improvement:

- Data Augmentation: Additional data augmentation techniques, such as random rotations or brightness adjustments, could help the model generalize even better.
- Transfer Learning: Leveraging a pre-trained model such as ResNet or VGG16 could provide a better starting point, potentially improving accuracy and reducing training time.
- Ensemble Methods: Combining predictions from multiple models could further enhance the robustness and accuracy of the final predictions.
- Class Imbalance Handling: Techniques such as oversampling the minority class or applying class weights could address any imbalances in the dataset, further improving the model's performance.

In future work, expanding the dataset, experimenting with more complex architectures, and exploring alternative deep learning techniques could push the boundaries of this project and provide even better tools for pneumonia detection.