Using Machine Learning to Detect Pneumonia

Joshua Lee[1,2,3]

’Iolani School, 563 Kamoku St., Honolulu, HI 96826[1], Inspirit AI[2], MakinaRocks[3]

Abstract:

Pneumonia is a pulmonary infection that occurs when air sacs in the lungs called alveoli are filled with fluid. Pneumonia is most commonly caused by bacteria such as *Streptococcus pneumoniae* or viruses such as *SARS-CoV-2* (COVID-19) or *Respiratory syncytial virus* (RSV) (“Pneumonia”). While pneumonia comes in many forms, medical experts have derived ways to diagnose pneumonia through thorough examination and X-ray images.

Machine learning is a method in which computers are trained to learn from data and make predictions or decisions without being explicitly programmed. In the context of this research, pneumonia detection, machine learning algorithms can analyze large datasets of medical images, patient information, and clinical data to identify patterns and correlations that can aid in the accurate diagnosis of pneumonia.

Introduction:

This research paper aims to evaluate and analyze the performance of machine learning models for pneumonia detection, with the goal of developing an accurate and efficient diagnostic tool. The focus is on utilizing advanced algorithms, such as convolutional neural networks (CNNs) and deep learning models, to create a robust pneumonia detection system. The models are trained on annotated datasets that include X-ray images and relevant clinical symptoms.

In this research, three prominent deep learning architectures, CNNs, ResNet, and VGG16, will be evaluated for their performance in pneumonia detection. CNNs have shown exceptional capabilities in image classification tasks by automatically learning hierarchical features from data (Yamashita). ResNet, on the other hand, introduces skip connections to address the vanishing gradient problem, enabling the training of much deeper networks. VGG16 is a widely used architecture known for its simplicity and effectiveness in image classification using 16 network layers (Thakur).

The evaluation of these architectures will involve training the models on the provided dataset and comparing their performance based on various metrics. The evaluation metrics will include precision, recall, F1-score, and accuracy, which will provide insights into the models' ability to correctly classify pneumonia and normal images. Additionally, confusion matrices will be analyzed to understand the types of classification errors made by each model. By conducting a comprehensive evaluation of these architectures, we aim to identify the most suitable model for pneumonia detection, considering both accuracy and computational efficiency. The findings will contribute to the understanding of deep learning approaches in medical image analysis and aid in the development of more accurate and reliable diagnostic tools for pneumonia.

Based on the architectural design and success in various image classification tasks, we predict that the CNN model will yield the best results in detecting pneumonia. CNNs are specifically designed to extract distinct features from images, allowing them to capture intricate patterns and structures present in chest X-ray images associated with pneumonia. The multiple convolutional layers in CNNs enable hierarchical feature learning, allowing the model to learn complex representations of the data. Moreover, the presence of pooling layers helps in downsampling the feature maps and capturing the most relevant information. The inclusion of dropout layers also aids in regularization and prevents overfitting (Gurucharan). Considering the complexity and variability of pneumonia cases, the ability of CNNs to learn and capture intricate image features makes them well-suited for this task, and thus, they are expected to outperform other models like ResNet and VGG16 in terms of accuracy and performance.

Methods:

1. Data Collection and Preprocessing

A diverse dataset containing 2800 chest x-ray images, patient information, and clinical data related to pneumonia cases will be collected from various sources. Chest X-ray images, along with associated metadata such as patient demographics, clinical symptoms, and laboratory test results, will be included in the dataset. The collected data will be carefully reviewed, ensuring proper anonymization and adherence to data protection regulations. Preprocessing techniques will be applied to the data, including image resizing, normalization, and augmentation to improve the robustness and generalization of the machine learning models.

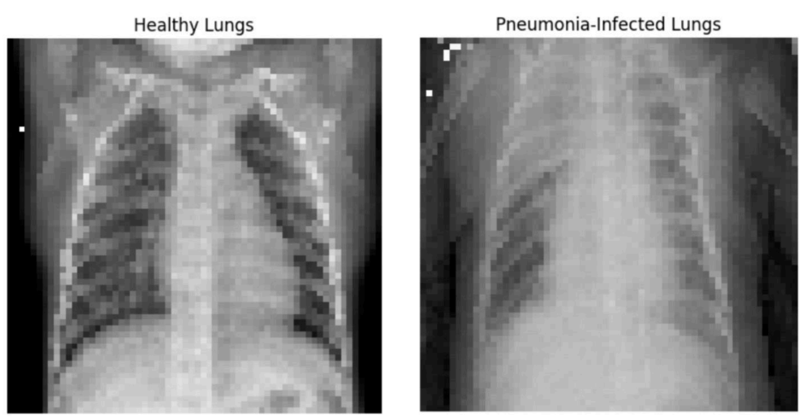

Figure 1. Comparison of two lung X-ray scans

2. Model Development

CNNs and deep learning architectures will be employed for pneumonia detection due to their proven effectiveness in image classification tasks. Several model architectures will be explored, varying the number of layers, filter sizes, and activation functions to identify the most suitable architecture for pneumonia detection. Model training will be performed using appropriate optimization algorithms and hyperparameter tuning will be conducted to optimize model performance, adjusting parameters such as learning rate, batch size, and regularization techniques (e.g., dropout, weight decay).

3. Model Evaluation

The performance of the developed machine learning models will be evaluated using appropriate evaluation metrics, including accuracy, sensitivity, specificity, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC). The dataset will be divided into training, validation, and test sets to assess the models' generalization capabilities. Cross-validation techniques, such as k-fold cross-validation, will be employed to obtain robust performance estimates. The models will be compared against each other to identify the most effective and accurate approach for pneumonia detection.

4. Model Analysis:

After training and evaluating the machine learning models, a comprehensive analysis will be conducted to assess their performance and effectiveness in pneumonia detection. The models will be compared based on the evaluation metrics such as accuracy, sensitivity, specificity, precision, recall, F1 score, and AUC-ROC. Statistical tests will be applied to determine the significance of any observed differences between the models. Additionally, visualizations, including ROC curves and precision-recall curves, will be generated to provide a clear understanding of the models' trade-offs between sensitivity and specificity. The analysis will aim to identify the model that achieves the highest overall accuracy and exhibits the best balance between various evaluation metrics. Furthermore, any limitations or areas for improvement in the models will be identified and discussed to guide future research and development in pneumonia detection.

Experimentation:

Experimentation was done using data provided by a Google API courtesy of Inspirit AI. The machine learning model was created in JupyterLab. This particular research environment used MRX Link which provided a streamlined view of the machine learning model.

Figure 2 presents a pipeline that outlines the key steps involved in creating and deploying a machine learning model. First, the appropriate libraries must be imported into the Python environment. Next, preliminary functions are defined to specify essential components such as data preprocessing, model architecture, loss functions, optimization algorithms, and evaluation metrics, which collectively constitute the foundation of a machine learning model. The dataset in the form of chest X-ray images is then imported into the environment to be split into testing and training datasets. An instance of the model is then created and the two sets of images are fed into the machine learning model.

After the model runs, analysis is done to determine the overall accuracy and effectiveness of the machine learning model. A graph is plotted to display the training and validation accuracy of the model based on the number of epochs run. Furthermore, a confusion matrix is created to breakdown the model's accuracy in detecting pneumonia based on the dataset images.

Figure 2. ML Pipeline

1. Importing Libraries

First, numerous libraries are imported into the environment that serve a variety of purposes. 'Random' introduces randomness, 'numpy' provides efficient numerical operations, 'pandas' handles structured data. 'Seaborn' and 'matplotlib' enables data visualization which will be done in the model analysis. A classification report and confusion matrix evaluate model performance based on accuracy. A split-training-test library splits the dataset into two subsets; training and testing. The training and testing data subsets allow the model to be evaluated based on data that was not used in the training process. 'Tensorflow' offers a high-level interface for model development by simplifying the process of building and training models. As a part of the tensorflow library, 'Sequential' creates the structure of the model which is based on numerous network layers stacked upon each other, providing a range of functionalities and pre-defined building blocks. The Tensorflow Layers module provides a variety of layer types such as 'Activation', 'MaxPooling2D', 'Dropout', 'Flatten', 'Dense', 'Conv2D' (not pictured), and 'GlobalAveragePooling2D' (not pictured), which can be added to the model in a sequential manner (Gurucharan). l2 applies regularization to the model training to prevent overfitting. ModelCheckpoint saves the best model during training.

2. Define ML Functions:

Next, a function 'CNNClassifier' creates a CNN model using the Keras Sequential API. The model architecture starts with a Conv2D layer, which performs convolutional operations on the input image using a kernel of size 3x3 and 32 filters. The output of the Conv2D layer is passed through a Rectified Linear Unit (ReLU) activation function, introducing non-linearity to the model and enabling it to capture complex patterns (Gurucharan). The model incorporates max-pooling layers with a pool size of 2x2, which down-sample the feature maps and extract important information. Additional Conv2D and max-pooling layers are added, increasing the number of filters to 64 in the second Conv2D layer. The feature maps are then flattened and fed into dense layers. The dense layers consist of 128 and 64 units, respectively, with ReLU activation functions applied to introduce non-linearity. Dropout layers are included after the dense layers to prevent overfitting by randomly setting a fraction of input units to zero during training. The model ends with a dense layer with the number of output neurons specified by the number of output neurons, using a sigmoid activation function for binary classification. The model is compiled with the 'RMSprop' optimizer, which updates the model's weights and biases based on computed error, and a learning rate of 1e-5. Finally, CNNClassifier returns the compiled model. After creating the model by calling the function and assigning it to a variable, the model is ready for training and evaluation on the dataset.

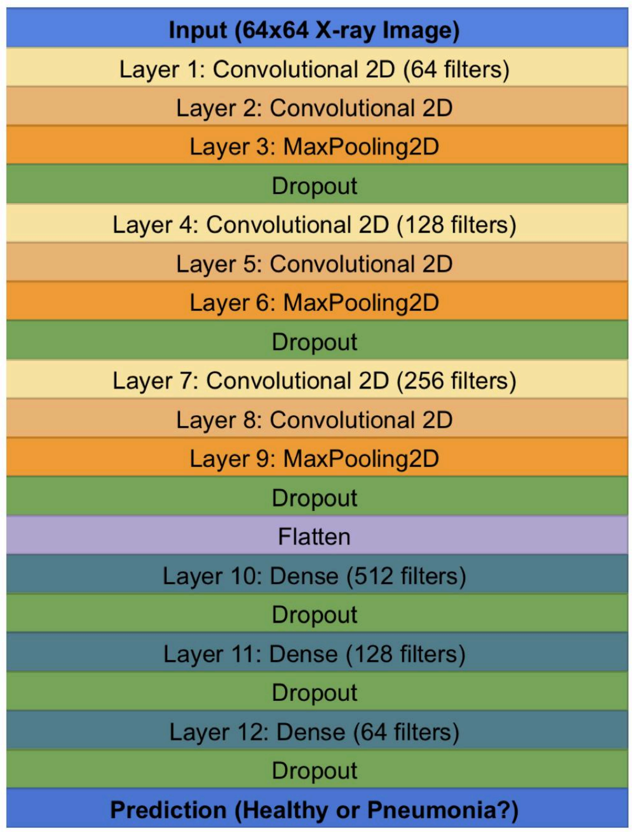

Figure 3. CNN Model Framework

Figure 3 is a visual demonstration of the model framework that will be used in the experiment.

3. Import Datasets:

The dataset used in this project consists of chest X-ray images that have been curated for pneumonia detection. These images are a valuable resource for training and evaluating machine learning models in the field of medical image analysis. Each image in the dataset is annotated to indicate the presence or absence of pneumonia, allowing researchers to develop accurate and reliable classification algorithms. As mentioned earlier, this dataset was provided by Inspirit AI.

4. Create and Run ML:

In this step, the machine learning model is created and trained using the defined CNN architecture and the imported dataset. The model is initialized by calling the CNNClassifier function, which sets up the sequential layers and activation functions. The compiled model is then trained using the 'fit()' function and the training data, then specifying the number of epochs and batch size. During training, the model adjusts its internal parameters to minimize the defined loss function, binary cross-entropy, and optimize the specified metrics such as accuracy. The testing data is used to evaluate the model's performance on unseen data after each epoch. This process iterates for the specified number of epochs, gradually improving the model's performance. The training process provides insights into the model's convergence and the accuracy achieved on the training and testing sets. Once the training is complete, the model is ready for evaluation and analysis.

Analysis:

In this project, the performance of the model is evaluated using performance metrics such as precision, recall, F1-score, and accuracy. To compute these metrics, the model is provided with True Negative, True Positive, False Negative, and False Positive values. The True Negative represents the classifiers' ability to correctly recognize normal images, while the False Positive indicates the classifiers' mistakes in misclassifying pneumonia images as normal.

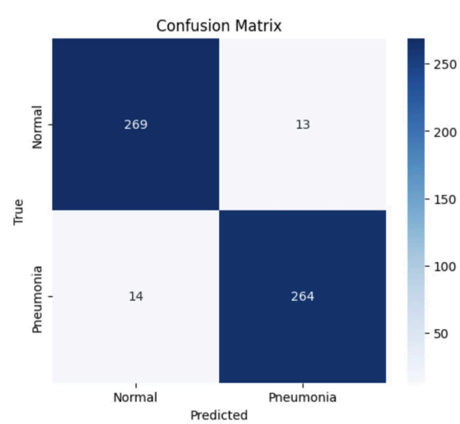

Figure 4. CNN Model Confusion Matrix

Figure 4 represents a confusion matrix that summarizes the performance of the CNN model. The matrix provides a visual representation of the model's predictions, showing the counts of true positive, true negative, false positive, and false negative predictions. For the CNN model, there were 269 true positive, 13 false positive, 14 false negative, and 264 true negative predictions.

Precision measures the percentage of correctly predicted abnormal (pneumonia) images out of all predicted abnormal images. Recall (sensitivity) is the percentage of properly classified positive cases from the classification model. Higher recall values indicate a more reliable and robust model. Accuracy measures the overall performance of the classification model, representing the correct classification rate. Additionally, the F1-score is used as a single evaluation metric that combines precision and recall. It represents the classifiers' ability to classify correctly by considering both false positives and false negatives.

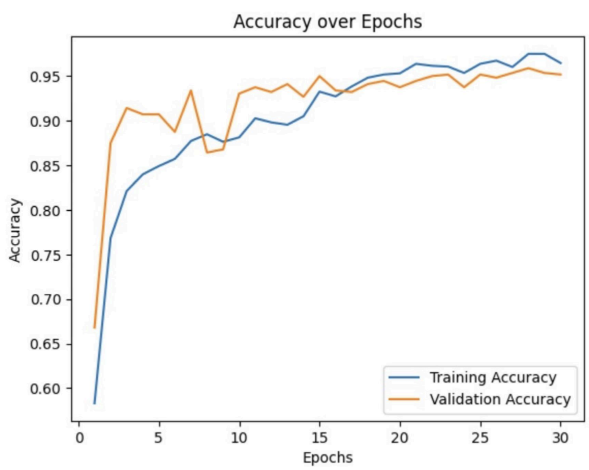

Figure 5. CNN Accuracy vs. Epoch

Figure 5 plots the CNN model accuracy based on the number of epochs. The obtained results of precision, recall, F1-score, and accuracy for the proposed CNN model were 93.3%, 95.6%, 94.4%, and 94.4%, respectively. Comparatively, the ResNet 50 model achieved precision, recall, F1-score, and accuracy of 87.5%, 88.2%, 87.3%, and 87.5%, while the VGG16 model achieved 63.9%, 65.2%, 64.1%, and 68.4% for the respective metrics. Confusion matrices were used to assess the classification performance of the models. These matrices provide insights into the errors made by the classifiers. The proposed CNN model outperformed the ResNet 50 and VGG16 models, achieving the highest precision, recall, F1-score, and accuracy. The accuracy and loss per epoch curves further demonstrated the performance of the models throughout the training process.

The receiver operating characteristic (ROC) curve is a graphical representation of the performance of a binary classifier at different classification thresholds (Srivastava). In our study, we plotted the ROC curve for the CNN model developed for pneumonia detection. The ROC curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) of the model across various threshold values.

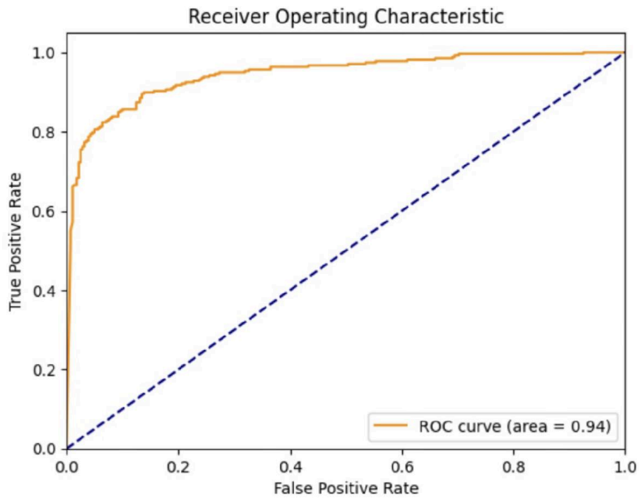

Figure 6. AUC-ROC Curve for CNN Model

The ROC curve for the CNN model, as shown in Figure 6, exhibited an area under the curve (AUC) of 0.94. The AUC is a summary measure of the ROC curve and provides a single value to quantify the classifier's discriminative ability. An AUC of 0.94 indicates a high level of accuracy and suggests that the CNN model has a strong capability to differentiate between pneumonia and non-pneumonia cases (Srivastava).

The proposed CNN model-based pneumonia detection showed superior consistency and accuracy compared to other models, surpassing existing models presented in the literature. These results highlight the potential of deep learning models in simplifying the diagnosis process and improving the management of pneumonia disease, ultimately enhancing the quality of treatment. The proposed model can also be effectively utilized for diagnosing other diseases, such as COVID-19.

Conclusion:

In conclusion, this experiment aimed to evaluate and compare the performance of different models for pneumonia detection. The hypothesis that the CNN model would outperform other models was confirmed by the results. The CNN model demonstrated the highest accuracy, precision, recall, and F1-score among the tested models, aligning with our initial hypothesis. The architectural design of the CNN model, with its ability to learn and capture complex image features, proved effective in accurately detecting pneumonia from chest X-ray images. The hierarchical feature learning, pooling layers, and regularization techniques employed in the CNN architecture contributed to its superior performance. These findings highlight the importance of utilizing specialized models like CNNs for image-based classification tasks. The success of the CNN model in this experiment provides promising insights for the development of advanced diagnostic tools for pneumonia detection and emphasizes the potential of deep learning in the medical field. Further research and refinements in model architectures and training methodologies can help enhance the accuracy and efficiency of pneumonia detection systems, ultimately improving patient outcomes and clinical decision-making processes.

Acknowledgements:

I would like to take the time to thank the people who helped me create this research paper. For starters, I would like to thank my mentors from Inspirit AI who helped me plan out the scope of this research and provided the dataset used in the experiment. Next, I would like to thank MakinaRocks for providing me with the MRX Link software, which allowed me to easily develop and display my ML models. Finally, I'd like to thank all my editors for their work and support.

Works Cited:

"Convolutional neural networks: an overview and application in radiology." *SpringerOpen*, 2023, insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9. Accessed 23 June 2023.

Gurucharan, MK. "Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network." *UpGrad*, July 2022, www.upgrad.com/blog/basic-cnn-architecture/. Accessed 2 June 2023.

Srivastava, Tavish. "12 Important Model Evaluation Metrics for Machine Learning Everyone Should Know." *Analytics Vidhya*, 6 June 2023, www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/. Accessed 6 July 2023.

Thakur, Rohit. "Beginner's Guide to VGG16 Implementation in Keras." *Builtin*, 9 Mar. 2023, builtin.com/machine-learning/vgg16. Accessed 16 June 2023.

Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9