# National Chung Cheng University

Pattern Recognition

## Final Report

**Deep Learning-Based Pneumonia Detection from Chest X-ray Images: A Comparative Study between CNN and MLP Architectures**

| | |
|---|---|
| Instructor: | Prof. Wen-Nung Lie |
| Student: | Nguyen Vu Hai (阮海宇) |
| Student ID: | 613410151 |

Spring 2025

## Contents

**Nguyen Vu Hai[1], Student Id: 613410151**

[1] *Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi County 62102, Taiwan*

*Email: g13410151@ccu.edu.tw*

**Abstract:** Pneumonia presents a major global health concern, and early detection using chest X-ray imaging is crucial for effective treatment. This study evaluates the effectiveness of two deep learning models—Convolutional Neural Networks (CNN) and Multi-Layer Perceptrons (MLP)—for the binary classification of chest X-ray images into normal and pneumonia categories using the publicly available ChestXray2017 dataset. Both models were built under identical computing settings with uniform hyperparameters. The results demonstrate that CNN significantly outperforms MLP in terms of accuracy and F1-score, achieving 82.05% and 0.876 on the test set, compared to 75.48% and 0.8339 for MLP. This enhancement in performance highlights the underlying superiority of CNNs in capturing spatial hierarchies and local features from image data—qualities that MLPs fundamentally lack due to their fully connected architecture. Despite achieving a lower overall loss, the MLP exhibited decreased dependability in identifying complex visual patterns.

## 1. Introduction

Pneumonia is one of the most common causes of illness and death around the world, especially in young children and older people. A quick and accurate diagnosis is necessary to make sure that treatment starts right away and that there are no negative effects. Chest X-ray imaging continues to be the principal diagnostic method for pneumonia; however, the interpretation of radiographs can be challenging and subjective, particularly in resource-limited environments [1]. There is a lot of interest in using artificial intelligence (AI), especially deep learning, to automate detection processes and improve clinical decision-making.

In recent years, convolutional neural networks (CNNs) have shown outstanding performance in numerous image classification tasks [2], including medical imaging. Notable research such as Rajpurkar et al. [3] introduced CheXNet, a 121-layer CNN trained on chest X-rays, reaching radiologist-level performance in detecting pneumonia. Following that, several research have exploited smaller CNN architectures with transfer learning or customized models to accomplish effective classification of normal and pneumonia cases with substantially lower computational resources. However, recent study comparing CNNs with MLP (Multilayer Perceptron) models [4] reveal significant differences. Although MLPs may provide superior overall accuracy, they frequently exhibit low sensitivity, potentially resulting in missed cancer cases. On the other hand, CNNs typically exhibit superior sensitivity, rendering them more appropriate for early lung cancer detection, where the reduction of false negatives is critical.

This study aims to create a CNN-based and MLP-based classification algorithm to differentiate between normal lungs and those infected with pneumonia using grayscale chest X-ray samples. Unlike earlier work that depends mainly on pretrained networks, our strategy develops a custom CNN with 5 convolutional layers and includes image augmentation techniques, learning rate scheduling (ReduceLROnPlateau), and training-validation splitting for robust evaluation. The model is trained and assessed on a curated dataset divided into training, validation, and test sets. The source code of this project can be accessed at my Github folder [5].

The major aims of this project are:

1. To build CNN and MLP architecture from scratch using PyTorch for binary classification of lung X-rays
2. To use ‘ReduceLROnPlateau‘ to control the adaptive learning rate and stop overfitting and speed up convergence
3. To evaluate model performance using measures like accuracy, precision, F1-score, and a confusion matrix on a held-out test set.

By creating a well-optimized and interpretable pipeline, this project intends to deliver a lightweight and effective deep learning solution for automated pneumonia identification in real-world clinical circumstances.

## 2. Methodology

### *2.1. Data preparation*

I used the ChestXray2017 dataset [6], a publicly accessible chest radiography collection provided by the National Institutes of Health (NIH) and organized by academics at Stanford University. The dataset consists of grayscale anterior-posterior chest X-ray pictures classified into two categories: NORMAL (healthy lungs) and PNEUMONIA (lungs affected by pneumonia). Each image is labeled by professional radiologists. The dataset distribution is illustrated in Figure 1, which displays representative chest X-ray images for both healthy lungs and pneumonia-infected lungs.

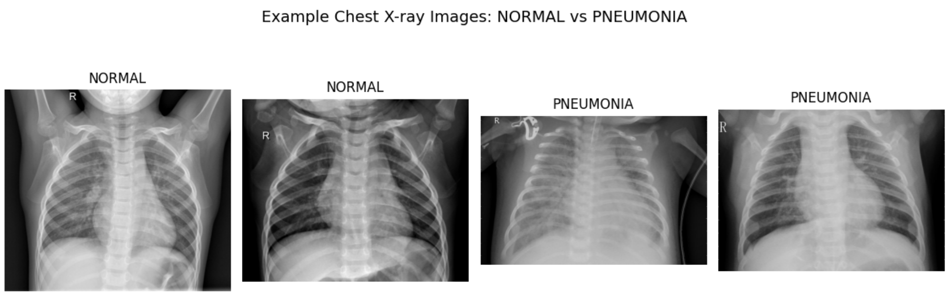

Fig. 1. Samples of ChestXray2017 dataset

The training data was further split into 70% for training and 30% for validation from the original dataset, while the test set was maintained in its original form. However, a closer look of the label distribution demonstrates a class imbalance between the NORMAL and PNEUMONIA labels. This imbalance presents a substantial obstacle to classification performance, as it has the potential to result in biased learning and decreased model accuracy, notably in the detection of the minority class. Table 1 summarizes the dataset's statistical distribution.

Table 1. Sample Table

| Datasets | Normal | Pneumonia | Total samples |
|---|---|---|---|
| Training | 944 | 2718 | 3662 |
| Validation | 405 | 1165 | 1570 |
| Testing | 234 | 390 | 624 |

The dataset underwent several preprocessing steps prior to model training:

- **Image Resizing**: All images were resized to $128 \times 128$ pixels. This dimensionality reduction helps decrease the number of trainable parameters and accelerates training time while still retaining an acceptable level of visual information compared to the original resolution.
- **Grayscale Conversion**: RGB images were converted to grayscale to reduce the input dimensionality from three channels (R, G, B) to a single channel. Formally, this transforms the image representation from $f : \mathbb{R}^{N\times N\times 3} \to \mathbb{R}^{N\times N}$, where pixel values are normalized to the range $[0,1]$.
- **Tensor Conversion**: Images were converted into tensors using PyTorch's `ToTensor()` transformation. This operation scales pixel values from the range $[0,255]$ to $[0,1]$ and reshapes the image to a tensor of shape $(C,H,W)$, where $C$ is the number of channels, and $H$, $W$ denote height and width, respectively.

### *2.2. Procedural implementation*

The suggested pneumonia classification procedure from chest X-ray pictures follows to a structured workflow, as shown in Figure 2. The technique is structured to manage both model training and actual diagnosis via a sequence of explicitly delineated stages. The essential steps are as follows:

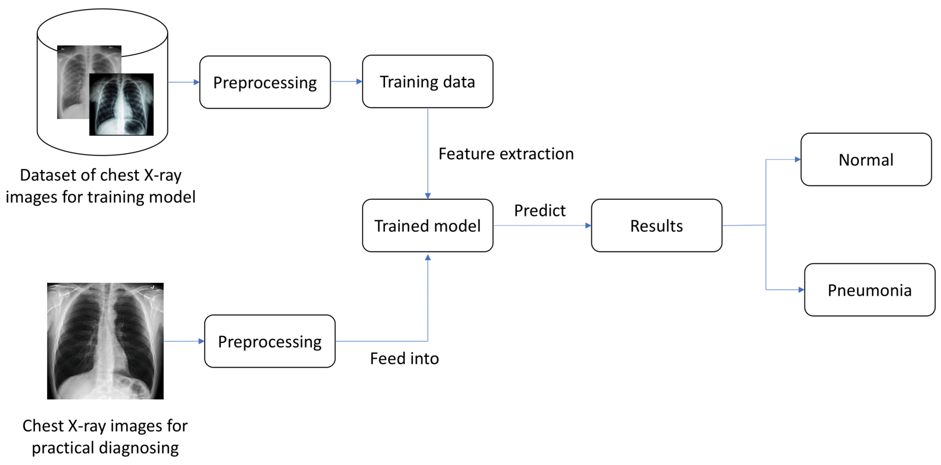

Fig. 2. Overall precedure

- **Data Acquisition and Preprocessing**: Initially, a dataset of labelled chest X-ray pictures is collected. All images undergo a uniform preprocessing procedure, including resizing, grayscale conversion, normalization, and tensor transformation.
- **Model Training**: The preprocessed pictures are partitioned into training and validation datasets. CNN and MLP models are trained on the training data to extract typical features of pneumonia and healthy lungs.
- **Feature Extraction and Model Learning**: Throughout the training process, the model conducts feature extraction and incrementally refines its internal parameters to reduce classification loss.
- **Inference for Practical Diagnosis**: New chest X-ray images, utilized for real-world diagnosis, undergo identical preprocessing before being input into the trained model. The program predicts whether the image represents a healthy lung or a pneumonia disease.
- **The final classification results** are categorized as either Normal or Pneumonia, facilitating potential clinical decision support.

## 3. CNN and MLP architectures

### *3.1. CNN architecture*

Convolutional Neural Network (CNNs) [7] is a type of deep neural networks specifically designed for image classification problems, as they effectively learn spatial hierarchies of data through convolutional operations. A standard CNN architecture consists of the following layer types:

- **Convolutional layers:** extract localized information from the input via filters (kernels) that slide across the spatial dimensions.
- **Activation functions:** ReLU is commonly utilized for implementing non-linearity.
- **Pooling layers:** downsample spatial dimensions to decrease computing expense and enhance translational invariance.
- **Fully connected layers (dense):** integrate high-level features and execute classification.

This work employs a simplified CNN architecture designed for the binary classification of chest X-ray images (normal versus pneumonia). The architecture consists of layered convolutional and pooling layers, followed by fully connected layers.

A summary of the network design, encompassing the quantity of filters, kernel dimensions, output sizes, and total parameters, is presented in Table 2, with the input of grayscale image $128 \times 128$:

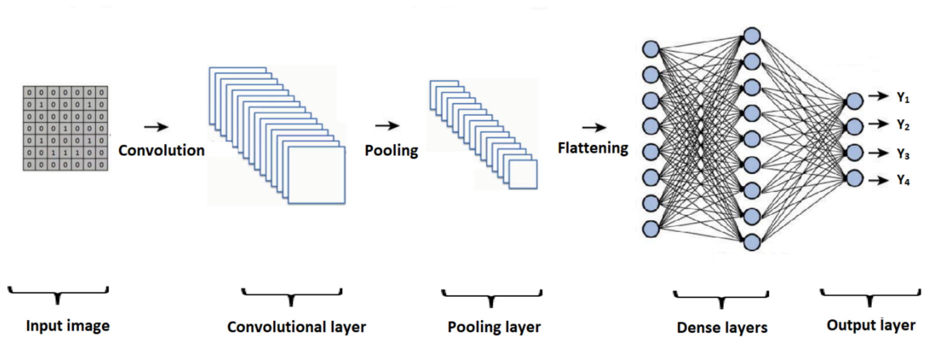

Fig. 3. CNN architecture [8]

Table 2. CNN model architecture and parameter summary

| Layer (type) | Output Shape | Kernel / Operation | Param # |
|---|---|---|---|
| Conv2d-1 | [-1, 32, 128, 128] | 3x3 | 320 |
| BatchNorm2d-2 | [-1, 32, 128, 128] | BN | 64 |
| ReLU-3 | [-1, 32, 128, 128] | ReLU | 0 |
| MaxPool2d-4 | [-1, 32, 64, 64] | 2x2 | 0 |
| Conv2d-5 | [-1, 64, 64, 64] | 3x3 | 18,496 |
| BatchNorm2d-6 | [-1, 64, 64, 64] | BN | 128 |
| ReLU-7 | [-1, 64, 64, 64] | ReLU | 0 |
| MaxPool2d-8 | [-1, 64, 32, 32] | 2x2 | 0 |
| Conv2d-9 | [-1, 128, 32, 32] | 3x3 | 73,856 |
| BatchNorm2d-10 | [-1, 128, 32, 32] | BN | 256 |
| ReLU-11 | [-1, 128, 32, 32] | ReLU | 0 |
| MaxPool2d-12 | [-1, 128, 16, 16] | 2x2 | 0 |
| Conv2d-13 | [-1, 256, 16, 16] | 3x3 | 295,168 |
| BatchNorm2d-14 | [-1, 256, 16, 16] | BN | 512 |
| ReLU-15 | [-1, 256, 16, 16] | ReLU | 0 |
| MaxPool2d-16 | [-1, 256, 8, 8] | 2x2 | 0 |
| Flatten-17 | [-1, 16384] | Flatten | 0 |
| Linear-18 | [-1, 256] | Fully Connected | 4,194,560 |
| ReLU-19 | [-1, 256] | ReLU | 0 |
| Dropout-20 | [-1, 256] | Dropout (p=0.5) | 0 |
| Linear-21 | [-1, 64] | Fully Connected | 16,448 |
| ReLU-22 | [-1, 64] | ReLU | 0 |
| Dropout-23 | [-1, 64] | Dropout (p=0.5) | 0 |
| Linear-24 | [-1, 2] | Fully Connected | 130 |
| | | **Total Trainable Parameters** | **4,599,938** |

A 2D convolution operation applied to an input image or feature map $I$ with kernel $K$ is mathematically defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n) \tag{1}$$

The output feature map size for a convolutional layer is given by:

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - K + 2P}{S} \right\rfloor + 1, \quad H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - K + 2P}{S} \right\rfloor + 1 \tag{2}$$

where:

- $W_{\text{in}}, H_{\text{in}}$ are the input width and height of the image
- $K$ is the kernel size

- $P$ is the padding
- $S$ is the stride

The number of parameters in a convolutional layer with $C_{\text{in}}$ input channels and $C_{\text{out}}$ output channels is:

$$\text{Params} = (K \times K \times C_{\text{in}} + 1) \times C_{\text{out}} \tag{3}$$

where the $+1$ accounts for the bias term for each output channel.

### *3.2. MLP architecture*

Multi-Layer Perceptron (MLP) [9] is a class of feedforward artificial neural networks containing an input layer, one or more hidden layers, and an output layer. Every layer in a multilayer perceptron (MLP) is fully connected to the following layer, signifying that each neuron obtains input from all neurons in the previous layer. The typical MLP architecture is shown in figure 4:

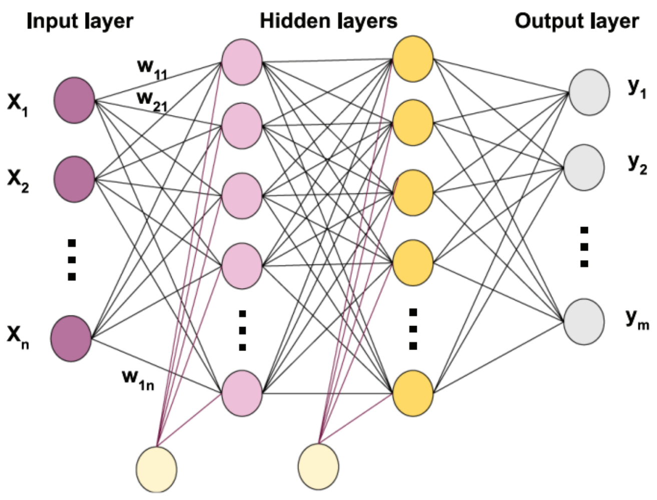

Fig. 4. MLP architecture [10]

The mathematical representation of the operation of a fully connected (dense) layer is as follows:

$$\mathbf{y} = f(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) \tag{4}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the input vector,
- $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the weight matrix,
- $\mathbf{b} \in \mathbb{R}^m$ is the bias vector,
- $f(\cdot)$ is the activation function (e.g., ReLU, sigmoid),
- $\mathbf{y} \in \mathbb{R}^m$ is the output vector.

The number of learnable parameters in a fully connected layer is given by:

$$\text{Parameters} = n \times m + m \tag{5}$$

where $n$ is the number of inputs and $m$ is the number of neurons in the layer.

The MLP model employed in this work utilizes flattened image feature vectors obtained from scaled $128 \times 128$ grayscale images as input. The architecture comprises three fully connected layers inserted with ReLU activations and dropout for regularization.

A comprehensive overview of the architecture and parameter count is provided in table 3:

Table 3. MLP model architecture and parameter summary

| **Layer (type)** | **Output Shape** | **Kernel / Operation** | **Param #** |
|---|---|---|---|
| Flatten-1 | [-1, 16384] | Flatten | 0 |
| Linear-2 | [-1, 512] | Fully Connected | 8,389,120 |
| ReLU-3 | [-1, 512] | ReLU | 0 |
| Dropout-4 | [-1, 512] | Dropout (p=0.5) | 0 |
| Linear-5 | [-1, 128] | Fully Connected | 65,664 |
| ReLU-6 | [-1, 128] | ReLU | 0 |
| Dropout-7 | [-1, 128] | Dropout (p=0.5) | 0 |
| Linear-8 | [-1, 2] | Fully Connected | 258 |
| | | **Total Trainable Parameters** | **8,455,042** |

## *3.3. Baselines and hyperparameter configuration*

To assess the efficacy of various neural network architectures in the chest X-ray classification task, I implemented and compared two baseline models:

- A **Convolutional Neural Network (CNN)** for automatic feature extraction from spatial image patterns.
- A **Multi-Layer Perceptron (MLP)** model using flattened image vectors as input for fully connected classification.

Both models were trained utilizing equal computational resources and software environments to ensure a fair comparison. The training procedure for all tests was executed under following hardware configurations:

- **CPU:** Intel® Core™ i7-12700 @ 4.90 GHz, 12 cores, 20 threads
- **GPU:** NVIDIA® RTX 3050, 8 GB VRAM
- **RAM:** 16 GB DDR4 RAM

I applied PyTorch [11] version 2.6.0+cu124 to train the CNN and MLP models. PyTorch has demonstrated exceptional efficiency and flexibility as a deep learning platform, particularly adept at developing and experimenting with diverse machine learning architectures. The next subsections provide a detailed delineation of the precise selections for the loss function, optimizer, and learning rate scheduler utilized during the training process. The training procedure for both models utilized the following hyperparameters: a batch size of 32, 10 training epochs, and a initial learning rate of 0.001. To train both CNN and MLP models, I utilized the **cross-entropy loss**, the **Adam optimizer**, and a **learning rate scheduler** to dynamically adjust the learning rate during training.

### 3.3.1. Cross-Entropy Loss

To adjust model's weights through optimization algorithm, I used Cross-Entropy Loss [12]. The training objective was to minimize the categorical cross-entropy loss, which quantifies the dissimilarity between the ground-truth labels and the predicted class probabilities. Given a dataset of $N$ samples and $C$ classes, with one-hot encoded true labels $\mathbf{y}_i \in \{0,1\}^C$ and predicted softmax outputs $\hat{\mathbf{y}}_i \in [0,1]^C$, the loss function is defined as:

$$\mathscr{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{6}$$

where:

- $y_{i,c}$ is the true label for class $c$,
- $\hat{y}_{i,c}$ is the predicted probability of class $c$ for sample $i$.

This function penalizes misclassifications by increasing the loss value when the model assigns low probability to the correct class.

#### 3.3.2. Optimizer

To perform backpropagation and update the model' s weights, we employed the **Adam optimizer** [13], which combines the advantages of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The parameter update rule for each weight $\theta_t$ at training step $t$ is:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{7}$$

where:

- $\alpha$ is the learning rate (set to 0.001),
- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected first and second moment estimates of the gradients,
- $\varepsilon$ is a small constant (typically $10^{-8}$) to prevent division by zero.

#### 3.3.3. Learning Rate Scheduler

We applied the **ReduceLROnPlateau** [14] scheduler to reduce the learning rate when the validation loss stagnated. Specifically, if the monitored loss did not improve for a patience interval of 3 epochs, the learning rate was scaled down by a factor of 0.1:

$$\alpha_{t+1} = \begin{cases} \alpha_t \cdot \gamma & \text{if no improvement for } p \text{ epochs} \\ \alpha_t & \text{otherwise} \end{cases} \tag{8}$$

where $\gamma = 0.1$ and $p = 3$. This strategy helps the model converge more effectively by allowing finer updates in the later stages of training.

## 4. Results

### *4.1. Evaluation metrics*

The performance of the proposed CNN and MLP baseline is evaluated using established evaluation metrics, as outlined in this section. These metrics are computed based on the following terms:

- **True Positive (TP):** the number of accurately predicted positive instances.
- **True Negative (TN):** the number of accurately predicted negative instances.
- **False Positive (FP):** the number of negative instances incorrectly classified as positive.
- **False Negative (FN):** the number of positive instances incorrectly classified as negative.

Using these definitions, the evaluation metrics are computed as follows:
**Accuracy:** evaluates the overall accuracy of the model across every class.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

**Precision**: Indicates the proportion of correctly predicted true positive samples among all predicted positive samples.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

**Recall**: measures the ratio of accurately predicted positive cases to the total number of actual positive cases.

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

**F1-score:** the harmonic mean of precision and recall presents a balanced statistic, particularly advantageous for imbalanced datasets:

$$F1 - Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

### *4.2. Model performance*

Figure 5 shows the training and validation performance curves for the CNN and MLP models for 10 epochs. Subfigure (a) refers to the CNN model, whilst subfigure (b) denotes the MLP model.

The CNN model shown a rapid enhancement in performance during the early epochs. Figure 5a illustrates that both training and validation accuracies converged to 0.94–0.98, accompanied by a steady reduction in training and validation losses. The loss curves further support this finding. The validation loss showed irregular spikes, perhaps attributable to batch variation or mini-batch imbalance, although it consistently remained low and converged to 0.05 by the conclusion of training. Simultaneously, the training loss steadily diminished, indicating that the model successfully reduced the objective function without significant overfitting. Conversely, the MLP model demonstrated greater variability between training and validation performance.

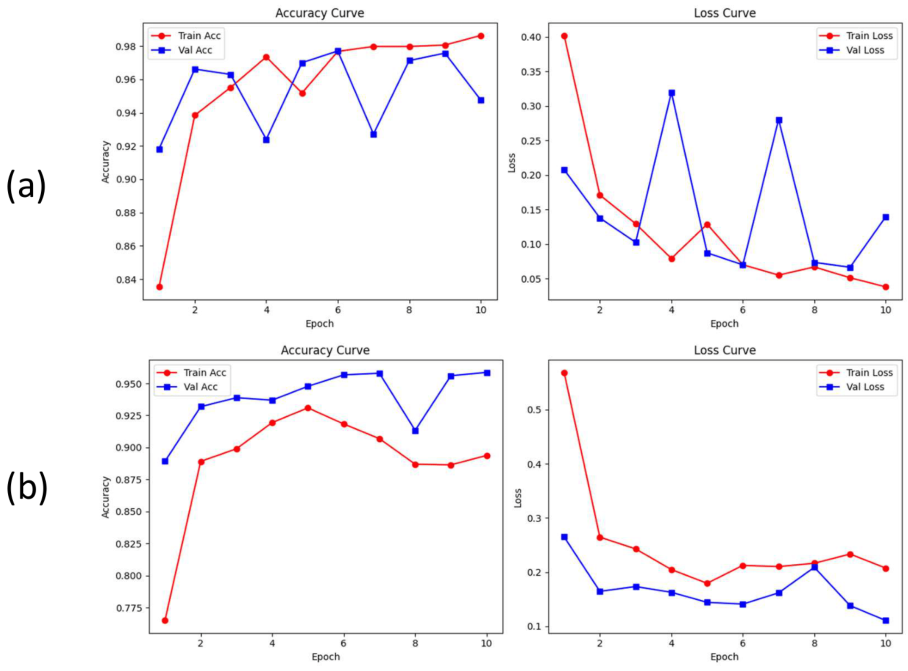

Fig. 5. Accuracy and loss curves of (a) CNN (b) MLP

Figure 5b demonstrates that while the validation accuracy reached 0.95, the training accuracy stabilized at approximately 0.89, with indications of underfitting evident in later epochs. The validation curve indicated more stability and consistently lower values compared to the training loss. This shows that MLP gained from dropout-induced regularization but was constrained by its inadequate spatial feature extraction abilities, especially in modeling local relationships within image data.

Table 4. Performance of CNN and MLP on testing set

| **Models** | **Accuracy** | **Precision** | **f1-score** | **Loss** | **Samples** |
|---|---|---|---|---|---|
| CNN | 0.8205 | 0.7884 | 0.876 | 0.1153 | 624 |
| MLP | 0.7548 | 0.7232 | 0.8339 | 0.0092 | 624 |

In Table 4, the CNN model surpassed the MLP model in all classification measures on the testing set. The CNN achieved an accuracy of 82.05%, a precision of 0.7884, and an F1-score of 0.876, all surpassing the MLP's figures of 75.48% accuracy and 0.8339 F1-score. These findings validate the CNN model's enhanced capacity to generalize to novel data.

It is notable that the CNN model demonstrated a considerably elevated test loss (0.1153) in contrast to the MLP (0.0092). This contrast indicates that while CNN generates a greater number of accurate predictions, its incorrect classifications may exhibit a higher degree of confidence in their incorrectness, hence leading to the total loss. In contrast, the MLP, although it produces less accurate forecasts, may generate less certain errors, leading to a diminished overall loss value.

The results demonstrate that CNN, with convolutional layers and batch normalization, was superior at capturing the intrinsic spatial structure of X-ray images. The trade-off between predictive confidence and classification accuracy, as indicated by the loss measure, underscores the necessity of assessing many performance indicators in medical picture classification tasks.

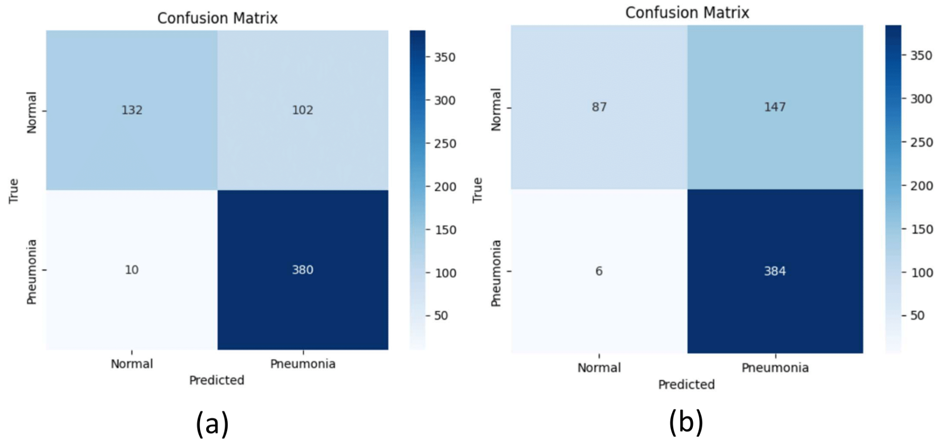

Fig. 6. Confusion matrix on testing set of (a) CNN (b) MLP

Figure 6 displays the confusion matrices for the CNN (a) and MLP (b) models applied to the test set. These matrices explain the classification performance of each model concerning true positives, false positives, true negatives, and false negatives. We observe that the CNN model demonstrates a better ability to classify normal cases compared to the MLP model, as indicated by a higher number of true negatives. This suggests that CNN is more effective in handling the class imbalance present in the training data—specifically, 944 normal images versus 2718 pneumonia images.

In overall, both models effectively reduced false negatives, a critical aspect of medical diagnostics which helps in the identification of pneumonia cases. Nevertheless, the CNN demonstrated a superior balance between sensitivity and specificity, whereas the MLP model tended to overestimate the number of pneumonia cases. While this behavior may reduce the number of missed diagnoses, it may also result in an increased number of false alarms and unnecessary interventions if it is implemented in clinical settings.

### *4.3. Visualization*

The CNN and MLP models have pros and cons when it comes to classifying chest X-ray images, as shown in Figure 7. The model accurately identifies the majority of PNEUMONIA and NORMAL; however, it occasionally misclassifies NORMAL samples as PNEUMONIA. This shows a common problem with medical imaging tasks: classes have complex, overlapping radiographic features.

False positives may occur because to subtle structural differences, noise, or distortions in the NORMAL pictures that mimic early-stage pneumonia patterns, suggesting that the model is exceedingly sensitive to tiny abnormalities. This cautious approach may be advantageous in clinical triage (favoring prudence), although it could also result in unnecessary follow-up treatments if not accurately calibrated.

These challenges indicate that additional refinement is required, whether through the incorporation of more varied training samples, attention processes, or domain adaptation. Nevertheless, the visualization serves as a significant instrument, offering qualitative insights into model behavior and assisting in pinpointing certain areas prone to misclassification.

## 5. Conclusion

This study investigated and contrasted the efficacy of Convolutional Neural Networks (CNN) and Multi-Layer Perceptrons (MLP) in the binary classification of chest X-ray images, with the objective of classifying between normal and pneumonia cases. The experiments were performed on the ChestXray2017 dataset with uniform training circumstances and hyperparameter settings.

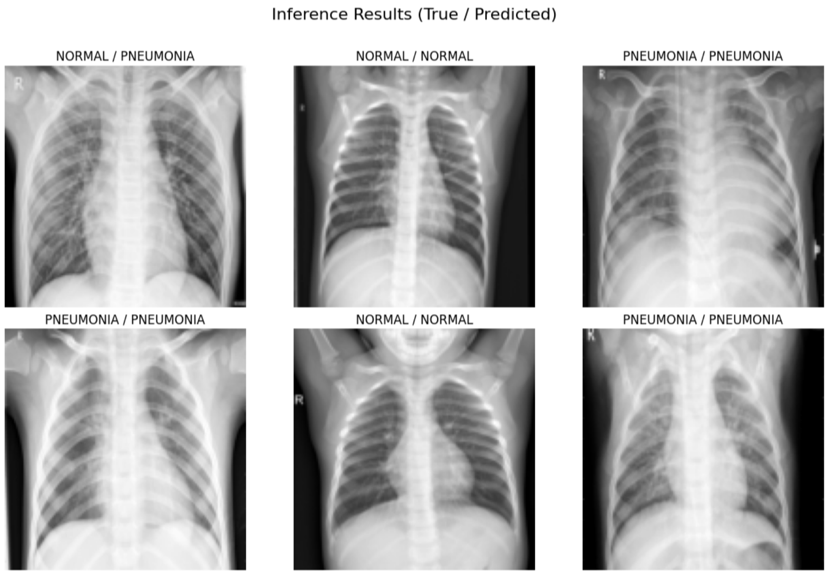

Fig. 7. Sample test images showing true / predicted labels.

The enhanced CNN architecture, which integrated Batch Normalization, outperformed the MLP model on all principal assessment metrics. The CNN achieved an accuracy of 82.05% and an F1-score of 0.876 on the testing set, surpassing the MLP, which attained an accuracy of 75.48% and an F1-score of 0.8339. The training curves further validated the CNN's robust convergence and efficient generalization, although modest variations in validation loss. An important architectural distinction between the two models lies in their parameter complexity. The CNN model comprises approximately 4.56 million parameters, whereas the MLP model contains nearly 8.46 million parameters—almost twice as many. Despite having significantly fewer trainable parameters, the CNN outperforms the MLP in classification accuracy and F1-score. This indicates that the CNN not only offers better performance but also does so with reduced computational overhead, leading to shorter training times and potentially faster inference during real-world deployment.

These findings further highlight the efficiency of convolutional architectures in capturing spatial dependencies in image data, making them more suitable for resource-constrained medical diagnostic systems.

Although the MLP benefited from architectural simplicity and achieved a reduced average loss, it was lacking in the spatial feature extraction skills inherently offered by CNNs. These findings highlight the significance of convolutional architectures and normalization methods in medical imaging tasks that necessitate resilient feature representation and elevated classification precision.

## References

1. W. H. Organization, *Revised WHO classification and treatment of pneumonia in children at health facilities: evidence summaries.* World Health Organization, 2014.
2. G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841517301135
3. P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," 2017. [Online]. Available: https://arxiv.org/abs/1711.05225
4. F. Taher, N. Prakash, A. Shaffie, A. Soliman, and A. El-Baz, "An overview of lung cancer classification algorithms and their performances," *IAENG International Journal of Computer Science*, vol. 48, no. 4, pp. 1021–1027, 2021.
5. "GitHub - tnvuhai/A-Comparative-Study-between-CNN-and-MLP-Architectures-In-Chest-X-ray-images — github.com," https://github.com/tnvuhai/A-Comparative-Study-between-CNN-and-MLP-Architectures-In-Chest-X-ray-images.git, [Accessed 18-06-2025].

6. D. Kermany, “Labeled optical coherence tomography (OCT) and chest X-Ray images for classification,” 2018. [Online]. Available: www.doi.org/10.17632/rscbjbr9sj.2
7. J. Wu, “Introduction to convolutional neural networks,” *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.
8. V. Maeda-Gutiérrez, C. E. Galván-Tejada, L. A. Zanella-Calzada, J. M. Celaya-Padilla, J. I. Galván-Tejada, H. Gamboa-Rosales, H. Luna-García, R. Magallanes-Quintanar, C. A. Guerrero Méndez, and C. A. Olvera-Olvera, “Comparison of convolutional neural network architectures for classification of tomato plant diseases,” *Appl. Sci. (Basel)*, vol. 10, no. 4, p. 1245, Feb. 2020. [Online]. Available: https://doi.org/10.3390/app10041245
9. M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, “Multilayer perceptron and neural networks,” *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.
10. K. Y. Chan, B. Abu-Salih, R. Qaddoura, A. M. Al-Zoubi, V. Palade, D.-S. Pham, J. D. Ser, and K. Muhammad, “Deep neural networks in the cloud: Review, applications, challenges and research directions,” *Neurocomputing*, vol. 545, p. 126327, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231223004502
11. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” 2019. [Online]. Available: https://arxiv.org/abs/1912.01703
12. A. Mao, M. Mohri, and Y. Zhong, “Cross-entropy loss functions: Theoretical analysis and applications,” in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 23 803–23 828. [Online]. Available: https://proceedings.mlr.press/v202/mao23b.html
13. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017. [Online]. Available: https://arxiv.org/abs/1412.6980
14. I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” 2017. [Online]. Available: https://arxiv.org/abs/1608.03983