**Evaluation of the Implemented Solution**

The AI-based pneumonia detection system developed in this project was rigorously evaluated on a labeled test dataset consisting of chest X-ray images. The following key evaluation metrics and insights were gathered using standard classification performance tools such as confusion matrix and classification report from scikit-learn:

* **Accuracy**: The model achieved high test accuracy, showcasing its ability to distinguish between *NORMAL* and *PNEUMONIA* cases effectively. This demonstrates the feasibility of applying deep learning for medical image diagnosis.
* **Classification Report**:
  + The model showed high precision and recall in detecting pneumonia, indicating low false positives and false negatives.
  + The F1-score balances these metrics, affirming that the model maintains a strong trade-off between precision and recall.
* **Confusion Matrix**:
  + A small number of misclassifications were observed, mostly between borderline cases.
  + This emphasizes the need for further enhancement using larger datasets or fine-tuned architectures for even better performance.

These results confirm that the model generalizes well and is suitable for aiding medical professionals in preliminary diagnosis.

**Documentation of Design and Paradigms Used**

**Problem Identification**

Pneumonia is a serious lung infection that requires timely detection. The challenge lies in the accurate and rapid identification of pneumonia from chest X-ray images. Misdiagnosis can lead to life-threatening complications, which makes automation in early detection critical.

**Solution Design**

The developed solution is an AI-powered diagnostic web app that allows users to upload chest X-ray images and receive instant diagnostic predictions. The project follows a modular structure incorporating multiple programming paradigms:

1. **Procedural Programming**:
   * Utilized in preprocessing.py, where the data loading and augmentation pipeline is written as a sequence of procedural steps.
   * The test.py and train.py scripts also use procedural logic to load, train, and evaluate models.
2. **Object-Oriented Programming (OOP)**:
   * The core classifier is implemented using OOP in model.py through the PneumoniaClassifier class.
   * It encapsulates model building, training, evaluation, and prediction functionalities, promoting code reuse and readability.
3. **Event-Driven and Functional Programming**:
   * The web interface in app.py is built using Streamlit, which relies on event-driven programming where user interactions trigger actions.
   * The use of asyncio in predict\_single.py for asynchronous execution exemplifies functional and concurrent programming constructs.

**Implementation Highlights**

* **Model Architecture**: A CNN model with multiple convolutional layers, dropout, and sigmoid output was used.
* **Preprocessing**: Images were normalized, augmented, and reshaped to a standard input size with grayscale channels.
* **Deployment**: A simple yet effective Streamlit UI allows users to interact with the model in real-time.

**Advantages and Limitations**

**Advantages**:

* High classification accuracy and real-time predictions.
* Intuitive and responsive web interface.
* Clean separation of concerns using different files for preprocessing, training, evaluation, and prediction.

**Limitations**:

* The current model only supports binary classification (*NORMAL* vs *PNEUMONIA*).
* Model performance may degrade on unseen data from different sources due to lack of generalization.
* Deployment is local and not hosted on a scalable cloud platform.