PROJECT 1 - IMAGE CLASSIFICATION

Deep Learning for Computer Vision

António Cruz (140129), David Isaac (120064), Erik Daskalyuk (120062), Ivan Magalhães (106586), Ricardo Pereira (120052)

Table of Contents

0. Project Scope and Objectives

Scope

This project focuses on developing an AI classification model to automatically analyze chest X-ray images and identify patients with a history of pneumonia. The solution leverages deep learning techniques to design, train and validate a convolutional neural network optimized for high recall, reflecting the clinical priority of not missing pneumonia cases.

Through systematic experimentation with the model architecture and training configuration, the project evaluates different design choices (such as the number of convolutional filters, the use of Batch Normalization, Dropout and L2 regularization, and the learning rate strategy) and selects a final model that performs reliably on a held-out test set. Particular emphasis is placed on understanding how the decision threshold affects recall, precision and the pattern of false negatives and false positives.

The system will:

- Process and classify chest X-ray images into two categories: Normal and Pneumonia.
- Enable the hospital to proactively identify at-risk patients for targeted awareness and preventive care in a retrospective screening setting.

Objectives

- Develop a baseline model Implement an initial convolutional neural network for binary classification of chest X-ray images, providing a reference point for subsequent improvements.
- Optimize model architecture and training configuration Refine the network structure (number of filters, pooling strategy, dense head) and training setup (optimizer, learning rate, class weights) to improve performance while keeping the model relatively simple and interpretable.
- Implement regularization and stabilization techniques Apply L2 regularization, Batch Normalization, Dropout, class weighting, early stopping and learning rate scheduling to reduce overfitting and improve generalization to unseen data.
- Evaluate and interpret model performance Assess the final model using metrics such as accuracy, AUC, PR-AUC, precision, recall and F1-score, analyze confusion matrices and threshold-dependent behavior, and justify the choice of the recommended operating threshold.
- Deliver a functional and documented solution Provide a well-documented Python codebase and a report detailing the dataset, methodology, model architecture, training process, evaluation results and rationale behind the selected model and threshold strategy.

Overall, the project aims to deliver a practical and recall-focused AI model that can assist healthcare professionals in the retrospective identification of patients with pneumonia history, supporting early preventive interventions and reduction of recurrence risk.

1. Introduction

This project addresses the healthcare challenge of identifying patients with a history of pneumonia from historical X-ray records, enabling targeted preventive care and reducing recurrence risk. Using Python and deep learning libraries, we developed a binary classification model explicitly optimized for high recall, prioritizing the detection of true pneumonia cases over minimizing false positives. This design aligns with the hospital's goal of secondary prevention, where missing vulnerable patients (false negatives) poses a greater risk than contacting healthy individuals (false positives).

The solution integrates data preprocessing, convolutional neural network design and systematic performance evaluation, with an emphasis on flexibility and robustness. In particular, the model is designed to support post-training threshold adjustments, allowing the hospital to choose operating points that balance recall and precision according to operational needs, from more comprehensive coverage to more resource-efficient outreach.

1.1 Environment Setup

The project uses the following key libraries:

- Scikit-learn: for evaluation and metric computation, including F1-score, confusion matrices and classification reports.
- Keras and TensorFlow: for model definition, training and optimization, including layers, regularizers, callbacks and custom metrics.
- Matplotlib, NumPy and Seaborn: for data visualization, analysis and exploratory data understanding.

In the following sections, we describe the dataset and preprocessing pipeline, then present the model architecture, training procedure, evaluation strategy and final conclusions.

2. Data Understanding

2.1 Dataset Overview

The Chest X-Ray Images (Pneumonia) dataset is a widely used medical imaging dataset curated for the detection and classification of pneumonia from chest X-ray images. It was originally made available by Daniel Kermany et al. (2018) on Kaggle and is derived from pediatric patients (aged approximately 1–5 years) treated at the Guangzhou Women and Children's Medical Center.

This dataset is intended to support research in medical image analysis, especially for developing and evaluating deep learning models for automated pneumonia detection. In this project, the images are converted to standardized grayscale inputs of 256×256 pixels, and separate subsets are constructed for training, validation and testing to allow for fair performance estimation on unseen data.

2.2 Data Description

The dataset contains chest X-ray images categorized into two primary classes:

- Normal: X-ray scans of patients with no radiological signs of pneumonia.
- Pneumonia: X-ray scans showing lung opacities consistent with pneumonia.

Within the Pneumonia label, the original dataset distinguishes two subcategories:

- Bacterial pneumonia
- Viral pneumonia

For the purposes of this project, these subcategories are treated as a single Pneumonia class, since the clinical objective is to identify patients with a history of pneumonia in general, regardless of the specific etiology.

2.3 Data Characteristics

- Format: JPEG chest X-ray images
- Classes: 2 (Normal, Pneumonia)
- Label encoding used in this project:

    1. 0 → Normal
    2. 1 → Pneumonia

All images are loaded in grayscale mode and resized to a fixed resolution of 256×256 pixels before being fed to the model, ensuring consistent input shape across the dataset.

2.4 Data Processing

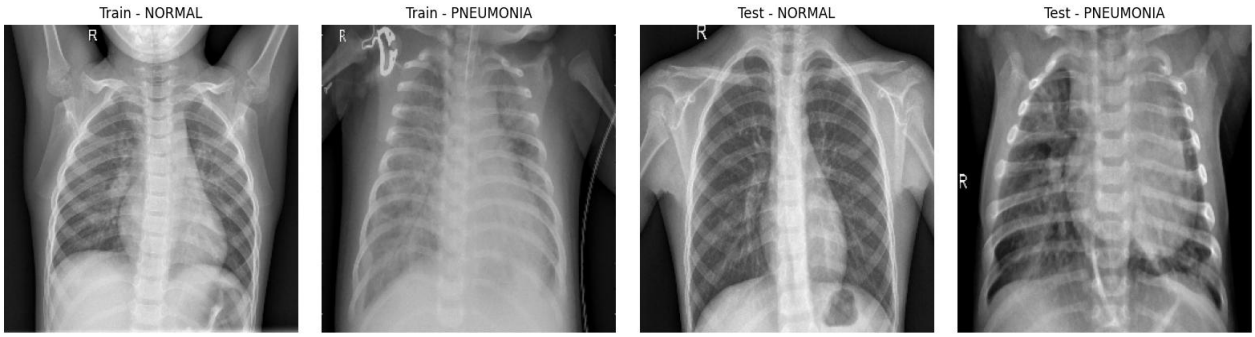

Figure 1 - Chest X-ray examples from the training and test sets (NORMAL and PNEUMONIA classes).

To gain a deeper understanding of the model's performance, it's essential to analyze the weight distribution of each class across the entire dataset.

In the following plot, the weight distribution across each class in the dataset is clearly shown. The dataset consists of 26% images of the NORMAL class and 74% images of the PNEUMONIA class.

This class imbalance could negatively impact model performance, as the model may become biased toward classifying images as PNEUMONIA, potentially leading to poorer performance on the NORMAL class.

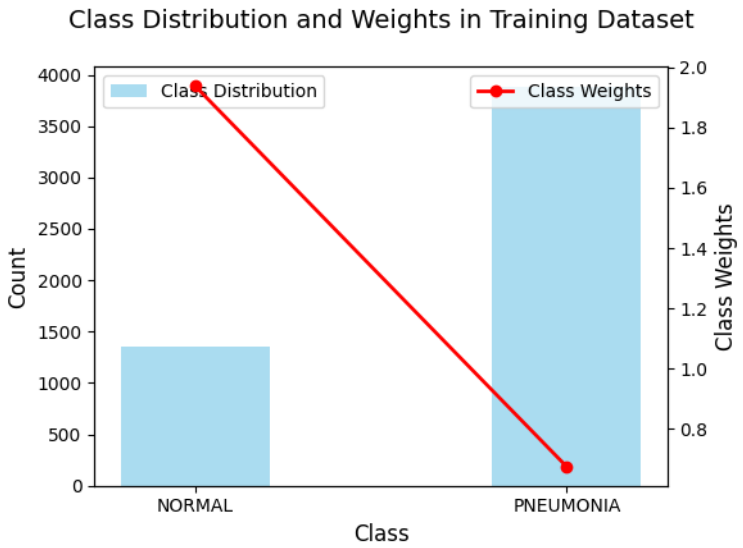

Figure 2 - Class distribution and corresponding training weights for NORMAL and PNEUMONIA in the training dataset.

In this step, we split the training data to create a validation set, allowing us to monitor and evaluate the model's performance throughout the training process.

We apply normalization to each dataset in order to enhance model generalization and improve its robustness during training.

We prepare the datasets for training and evaluation by batching and prefetching the data using the `tf.data` API.

- Batching: The training, validation and test datasets are batched with the chosen `batch_size`, so that examples are grouped into manageable chunks for efficient processing on the GPU/CPU. In this step:
    - `train_for_fit_dataset`, `val_dataset_gray` and `test_dataset_gray` are all converted into batched datasets.
- Prefetching: To improve performance, we call `.prefetch(tf.data.AUTOTUNE)` on each dataset. This allows the input pipeline to prepare the next batch in the background while the model is training or evaluating on the current batch, reducing idle time and improving throughput.

After batching and prefetching, we validate the setup by inspecting one batch from the training and validation datasets, checking the batch shapes and the range of pixel values. This confirms that the data is being fed to the model with the expected dimensions and normalization.

3. Training

3.1 Training Configuration and Callbacks

In this section, we build and train a Convolutional Neural Network (CNN) to classify chest X-ray images into two categories: Normal and Pneumonia. The model is trained on the preprocessed grayscale dataset using the AdamW optimizer, L2 regularization, Batch

Normalization and Dropout. To improve generalization and avoid overfitting, we use three Keras callbacks: ModelCheckpoint, EarlyStopping and ReduceLROnPlateau.

ModelCheckpoint and EarlyStopping monitor the validation PR-AUC (`val_pr_auc`), which is well suited for imbalanced problems and directly reflects the balance between precision and recall. ReduceLROnPlateau monitors the validation loss (val_loss) and reduces the learning rate when progress stalls, helping the optimizer make finer updates later in training.

The main callback parameters used in this project are summarized below.

Table 1 – Model callback parameter descriptions

Parameter	Meaning
`monitor`	Metric being tracked to decide whether there is an improvement. In this project: `val_pr_auc` for model checkpointing and early stopping, and `val_loss` for learning rate scheduling.
`patience`	Number of epochs to wait without improvement in the monitored metric before stopping or reducing the learning rate.
`mode`	Direction of optimization for the monitored metric. For `val_pr_auc` we use `"max"` (higher is better); for `val_loss` we use `"min"` (lower is better).
`save_best_only`	When `True`, only saves the checkpoint corresponding to the best observed value of the monitored metric.
`save_weights_only`	When `False`, saves the full model (architecture and weights) instead of just the weights.
`verbose`	Controls how much logging is printed during training when the callback is triggered.
`restore_best_weights`	When `True`, restores the model weights from the epoch with the best monitored metric after early stopping.
`min_lr`	Minimum learning rate allowed by the scheduler; prevents the learning rate from being reduced too far.
`factor`	Multiplicative factor used to reduce the learning rate when the monitored metric stops improving.

3.2 Model Checkpoint Saving

We set up a callback to save the best model during training based on validation PR-AUC rather than raw accuracy. This is more appropriate for an imbalanced, clinically oriented problem where the balance between precision and recall is more important than overall accuracy.

The ModelCheckpoint callback:

- Saves the model to the file path defined by `CHECKPOINT_PATH`.
- Monitors the `val_pr_auc` metric on the validation set.
- Keeps only the model corresponding to the highest observed `val_pr_auc` during training.

By saving the best validation-PR-AUC checkpoint instead of simply using the weights from the final epoch, we reduce the risk of overfitting. If performance on the validation set starts to degrade after a certain point, we still retain the version of the model that showed the best generalization according to the chosen metric.

3.3 Early Stopping

We also use an EarlyStopping callback to automatically stop training when further epochs no longer improve the model's performance on the validation set.

In this project:

- EarlyStopping monitors the validation PR-AUC (`val_pr_auc`), not the training metrics.
- If `val_pr_auc` does not improve for a specified number of epochs (the patience value), training is stopped.
- With `restore_best_weights=True`, the model weights are rolled back to those from the epoch with the highest `val_pr_auc`.

This prevents wasting epochs once the model has effectively converged and reduces overfitting by avoiding continued training after validation performance has stopped improving, while still keeping the best-performing version of the model for subsequent evaluation.

3.4 Learning Rate

The learning rate is a key hyperparameter that controls how much the model weights are updated during training. If it is too high, the optimizer may overshoot good minima and converge to a suboptimal solution; if it is too low, training can become very slow or get stuck.

In this project, we use the AdamW optimizer with an initial learning rate defined by `LEARNING_RATE`, together with a learning rate scheduler based on the `ReduceLROnPlateau` callback. The scheduler:

- Monitors the validation loss (`val_loss`).
- If `val_loss` does not improve for a certain number of epochs (patience), it reduces the learning rate by a given factor.
- Ensures that the learning rate never goes below a specified minimum (`min_lr`).

This strategy allows the model to make relatively large updates in the early stages of training, then gradually switch to smaller, finer updates once progress slows. In practice, this helps the model converge more reliably and can improve final validation performance without manual tuning of the learning rate schedule.

3.5 Build and Compile

The CNN model is created using the Keras Sequential API to perform binary classification of 256×256 grayscale chest X-ray images into Normal and Pneumonia. It combines three convolutional blocks, global average pooling and a small dense head with L2 regularization, Batch Normalization, Dropout and AdamW optimization. The main architectural and training choices are summarized in Table 2.

Table 2 - Summary of the CNN architecture, training configuration and optimization components

Component	Description
Input layer	Accepts grayscale images of size 256×256 pixels, with input shape (256, 256, 1).
Convolutional layers	Three convolutional layers with 64, 128 and 256 filters, respectively, using 3×3 kernels and ReLU activation. Each convolutional layer is followed by Batch Normalization and a

	MaxPooling2D layer to reduce spatial dimensions and improve training stability.
Global average pooling	A GlobalAveragePooling2D layer reduces the final feature maps to a single feature vector per image by averaging each feature map. This greatly reduces the number of parameters compared with Flatten, helps prevent overfitting and preserves the most salient information from each channel.
Dense layers	A fully connected layer with 128 neurons and ReLU activation, followed by a Dropout layer with rate 0.33, and an output layer with 1 neuron and sigmoid activation for binary classification (probability of Pneumonia).
Regularization	L2 regularization is applied to the convolutional and dense layers. Batch Normalization after each convolutional layer and Dropout after the dense layer further help to stabilize training and reduce overfitting.
Optimizer	Uses the AdamW optimizer with a specified learning rate and weight decay to improve convergence and regularization of the weights.
Metrics	During training and validation the model is monitored using accuracy, AUC, PR-AUC, precision and recall.
Callbacks	Includes model checkpointing (monitoring validation PR-AUC), early stopping (also on validation PR-AUC) and learning rate scheduling (monitoring validation loss) to adapt the learning rate during training and prevent overfitting.
Training	The model is trained on the training dataset, validated on a separate validation set, and class weights are used to compensate for the imbalance between Normal and Pneumonia cases.
Final result	The training process is monitored with the callbacks, and the model checkpoint with the best validation PR-AUC is saved for later evaluation on the held-out test dataset.

4. Evaluation

In this section, the final model is evaluated on a held-out test set using clinically relevant classification metrics (accuracy, precision, recall, F1-score and AUC), together with confusion matrices, Precision–Recall analysis and threshold-based error summaries. We also inspect the loss and accuracy curves over epochs to verify that training converged without severe overfitting before selecting the best checkpoint for testing.

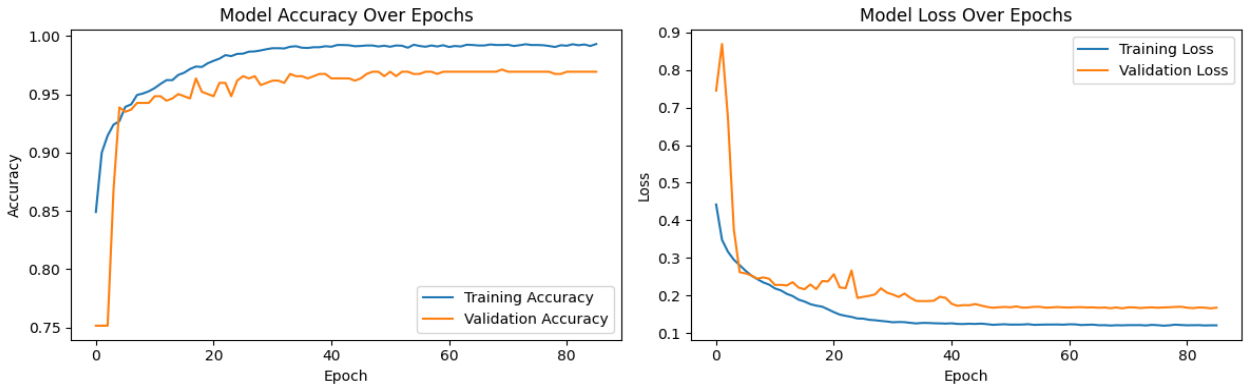

Figure 3 - Training and validation accuracy and loss curves over epochs for the CNN model.

During training, the model's learning curves show that it is able to fit the task without obvious instability. Training accuracy increases steadily while training loss decreases, and the validation curves, although noisier, improve in the same direction before stabilizing. The early-stopping and checkpoint callbacks, driven by the validation PR-AUC metric, prevent the model from overfitting and select the epoch that offers the best balance between fit and generalization.

When the selected checkpoint is evaluated on the held-out test set using the *default* decision threshold (0.5), the following metrics are obtained (F1-score computed from the reported precision and recall at threshold 0.5):

Table 3 - Test set performance of the selected model (threshold = 0.5).

Metric	Value
Accuracy	0.7949
Loss	0.9249
AUC	0.8979
PR-AUC	0.8959
Precision	0.7529
Recall	1.0000
F1-score*	0.8590

These values summarize the baseline test performance before considering alternative operating thresholds, which are analyzed in detail in the following subsections.

On the held-out test set, the model achieves solid baseline performance at the default decision threshold of 0.5, with accuracy of 0.7949, AUC of 0.8979 and PR-AUC of 0.8959. The detailed classification report at this threshold shows that performance is strongly asymmetric between the two classes, reflecting both the class imbalance and the recall-focused behavior of the model:

Table 4 – Detailed test performance by class (*default* threshold = 0.5)

Class	Precision	Recall	F1-score	Support
Normal	1.0000	0.4530	0.6235	234
Pneumonia	0.7529	1.0000	0.8590	390

At this operating point, all 390 pneumonia cases in the test set are correctly identified (recall 1.0000), but this comes at the cost of misclassifying 128 out of 234 Normal cases as Pneumonia, which explains the relatively low recall for the Normal class. This behavior is consistent with the clinical objective of prioritizing sensitivity to pneumonia over perfectly preserving specificity in Normal cases.

Because the retrospective screening program values missing as few pneumonia patients as possible more than avoiding outreach to healthy patients, the primary focus of evaluation is on recall and the pattern of false negatives versus false positives. To systematically explore this trade-off, we varied the classification threshold from 0.1 to 0.9 and computed accuracy, precision, recall, F1-score and confusion matrices at each point. The resulting threshold analysis, summarized in the next section, shows a wide range of thresholds where recall remains at 100% and motivates the recommended operating threshold of 0.6 used in the Conclusions.

5. Conclusions

5.1 Key Findings

The hospital's objective is to identify patients from historical X-ray records who had pneumonia, in order to contact them for preventive care and education to reduce recurrence risk. This is a retrospective screening program for secondary prevention.

In this context, the consequences of classification errors are highly asymmetric:

- A False Negative (FN) occurs when a patient who had pneumonia is not identified. This is a high-consequence error: the patient is missed and receives no preventive outreach, leaving them at elevated risk for recurrence, complications, or hospitalization.
- A False Positive (FP) occurs when a healthy patient is incorrectly flagged. This is a low-consequence error, resulting in minor administrative overhead and a patient receiving generally useful health information.

This asymmetry dictates our optimization strategy: we must prioritize high recall (catching true pneumonia cases) over high precision. Missing vulnerable patients is far more problematic than contacting some healthy individuals. During training, model checkpoints were selected using the validation PR-AUC metric, which balances precision and recall under class imbalance, and the final operating characteristics were analyzed in detail across multiple thresholds with particular attention to recall.

The model outputs probability scores (0 to 1). The classification threshold determines the cutoff for classification:

- Higher threshold (e.g., 0.7–0.9): more conservative, fewer patients flagged, but potentially more missed pneumonia cases.
- Lower threshold (e.g., 0.1–0.3): more inclusive, more patients flagged, typically fewer or no missed pneumonia cases but more unnecessary contacts.

5.1.1 Results Across Different Thresholds

Based on the test set evaluation (624 total samples: 390 Pneumonia, 234 Normal), the model demonstrates the following trade-offs:

Table 5 - Recall-precision trade-offs and error counts across decision thresholds on the test set

Threshold	Recall (Sensitivity)	Precision	Missed Cases (FN)	Unnecessary Contacts (FP)	Trade-off
0.1	100.0% (390/390)	68.3%	0	181	Very low threshold; maximum coverage, many unnecessary contacts
0.2	100.0% (390/390)	70.3%	0	165	Very low threshold; maximum coverage, many unnecessary contacts
0.3	100.0% (390/390)	72.2%	0	150	Inclusive; all pneumonia cases found, high number of unnecessary contacts
0.4	100.0%	73.2%	0	143	Inclusive; all pneumonia cases found, slightly fewer unnecessary

	(390/390)				contacts
0.5	100.0% (390/390)	75.3%	0	128	Balanced; all pneumonia cases found, fewer unnecessary contacts
0.6	100.0% (390/390)	76.5%	0	120	More conservative; all pneumonia cases found, fewer unnecessary contacts
0.7	99.7% (389/390)	78.9%	1	104	Conservative; one missed pneumonia case, fewer unnecessary contacts
0.8	99.0% (386/390)	79.9%	4	97	Very selective; a few missed cases, further reduction in unnecessary contacts
0.9	97.4% (380/390)	82.6%	10	80	Extremely selective; more missed cases, substantially fewer unnecessary contacts

5.1.2 Interpretation of Threshold Trade-offs

Based on this data, our insights from the evaluation can be summarized as follows:

1. **There is a wide plateau where recall remains at 100%.**
   - **a.** For thresholds between 0.1 and 0.6, the model achieves 100% recall (390 out of 390 pneumonia cases correctly identified) regardless of the exact threshold.
   - **b.** Within this region, increasing the threshold mainly reduces unnecessary contacts. For example, raising the threshold from 0.3 to 0.6:
     - i. Missed cases (FN): 0 → 0 (no change; all pneumonia cases are still found).
     - ii. Unnecessary contacts (FP): 150 → 120 (30 fewer healthy patients unnecessarily contacted).
2. **Above roughly 0.7, we begin to trade recall for fewer unnecessary contacts.**
   - **a.** Moving from 0.6 to 0.9:
     - i. Missed cases (FN): 0 → 10 (10 pneumonia patients no longer identified).
     - ii. Unnecessary contacts (FP): 120 → 80 (40 fewer healthy patients contacted).
   - **b.** This illustrates the classic trade-off: higher thresholds reduce outreach workload but introduce a small number of missed pneumonia cases.
3. **The recommended performance (threshold = 0.6) is strong and recall-focused.**
   - **a.** Recall: 100.0% – the model correctly identifies 390 out of 390 pneumonia cases, missing none.
   - **b.** Precision: 76.5% – roughly three out of four flagged patients truly had pneumonia in the historical data.
   - **c.** Unnecessary contacts: 120 healthy patients (out of 234 Normal cases) would be contacted, while 114 healthy patients would not be contacted.

This recommended operating point offers full coverage of pneumonia cases on the test set, with a substantial reduction in unnecessary contacts compared to contacting everyone, while still leaving room to adjust the threshold if the hospital wishes to further reduce outreach volume at the cost of a small number of missed cases.

5.1.3 Confusion Matrices Across Thresholds

Figure 4 shows a grid of confusion matrices for thresholds from 0.1 to 0.9 on the test set. This visualization makes it possible to see how the balance between false positives and false negatives changes as the threshold is adjusted.

The panel labeled “Threshold = 0.6 (recommended)” corresponds to the operating point discussed above. At this threshold, all 390 pneumonia cases are correctly identified (0 missed), and among 234 normal cases, 114 are correctly not contacted and 120 are unnecessarily contacted. These counts match the values reported for threshold 0.6 in Table 5.

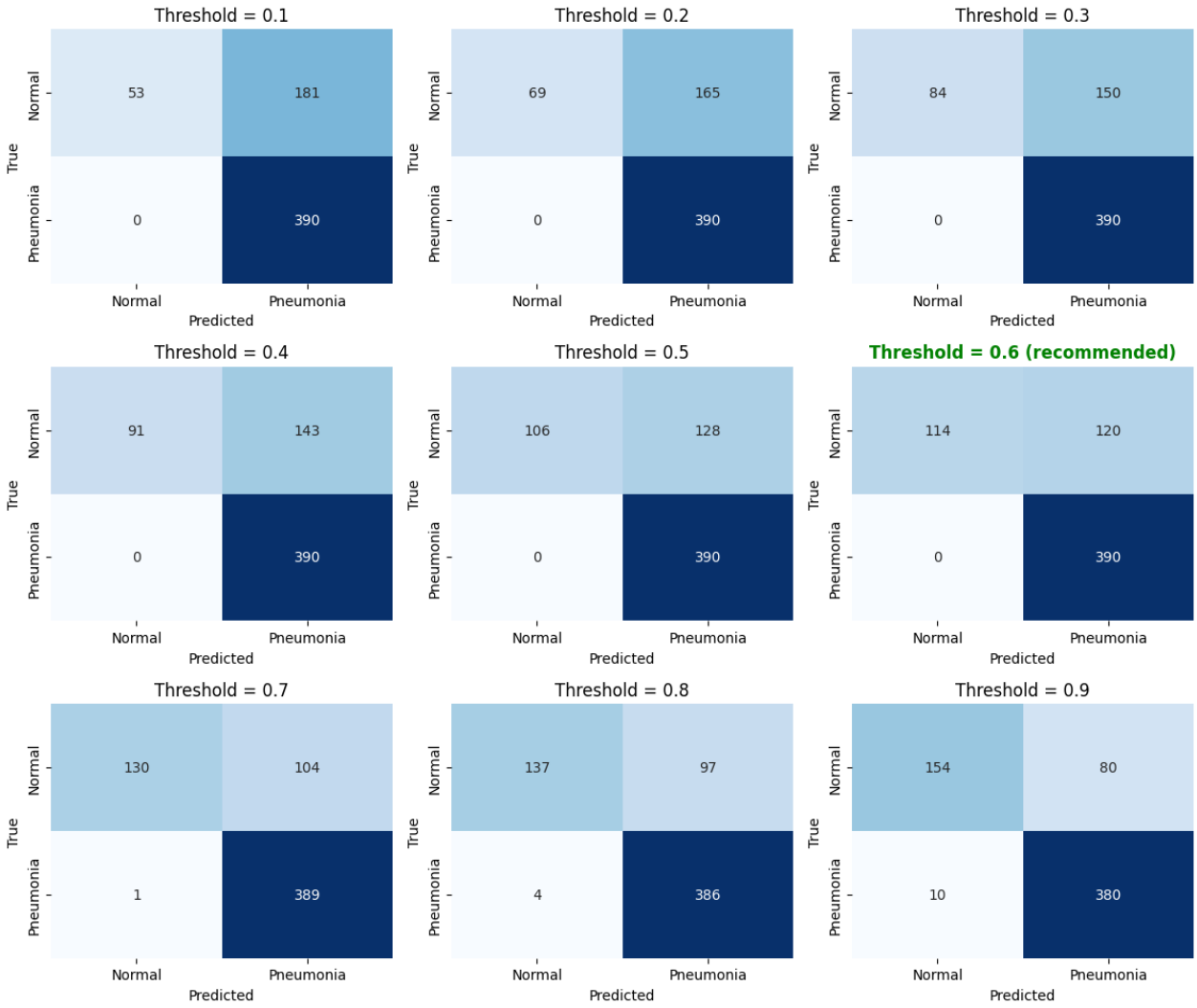

Figure 4 - Confusion matrices on the test set for thresholds (0.6 shown as recommended threshold).

5.1.4 Effect of Threshold on Global Metrics

Figure 5 summarizes how recall, precision and F1-score evolve as the decision threshold varies from 0.1 to 0.9. Recall remains at 100% across a broad plateau between thresholds 0.1 and 0.6, confirming that all pneumonia cases are captured in this range regardless of the exact cutoff.

Within this plateau, increasing the threshold gradually improves precision and F1-score by reducing the number of unnecessary contacts, without sacrificing coverage of pneumonia cases. Beyond approximately 0.7, recall starts to decrease as some pneumonia cases are

missed, while precision continues to improve. The vertical dashed line marks the recommended threshold of 0.6, which lies at the upper end of the full-recall region and provides the best balance between maintaining 100% recall and minimizing false positives.

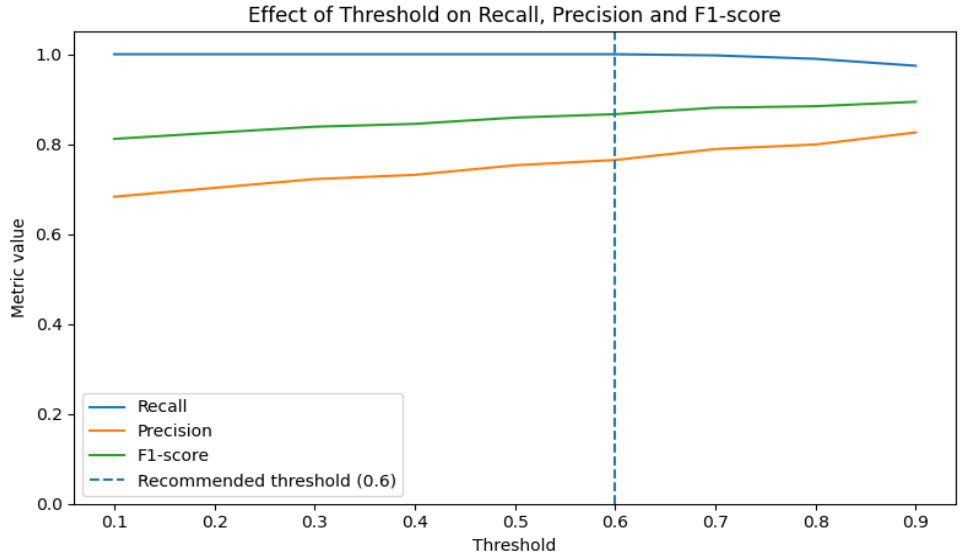

Figure 5 - Recall, precision and F1-score on the test set as a function of threshold.

5.2 Limitations

The Precision–Recall curve and the threshold–metrics plot highlight the trade-off between recall and precision when choosing the decision threshold. In general, moving along the curve means exchanging fewer missed pneumonia cases for more unnecessary contacts, or vice versa. In our specific model, there is a wide plateau (thresholds 0.1 to 0.6) where recall remains at 100%, and only beyond approximately 0.7 does recall begin to decrease.

- Lower thresholds (right side of the PR curve, higher recall): fewer missed patients but more unnecessary contacts (lower precision).
- Higher thresholds (left side of the PR curve, lower recall): fewer unnecessary contacts (higher precision) but more missed patients, once we move beyond the full-recall region.

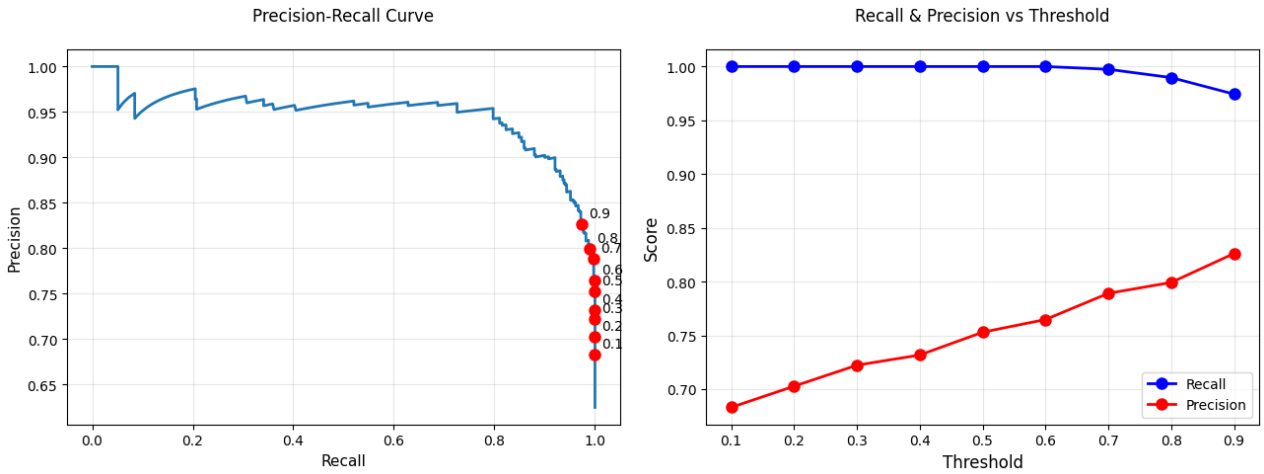

Figure 6 - Precision–recall curve and recall/precision vs threshold on the test set.

5.3 Recommendations

The trained model enables post-training business decisions based on resource availability and risk tolerance. Given the high cost of false negatives, the primary constraint is to operate in a region where recall remains as close to 100% as possible. On the test set, thresholds between 0.1 and 0.6 all achieve 100% recall (390 out of 390 pneumonia cases correctly identified).

Among these, threshold 0.6 yields the lowest number of false positives (120), with precision of 76.5% and no missed pneumonia cases, and is therefore recommended as the standard operating point.

Based on these conclusions, a simple decision framework can be proposed, summarized in the following operational scenarios:

Scenario 1 - Conservative outreach (limited resources)

- Threshold: **0.8**
- Strategy: Contact only higher-confidence cases, accepting a small number of missed pneumonia patients in exchange for fewer unnecessary contacts.
- Outcome: 99.0% recall (386 out of 390 pneumonia cases correctly identified, 4 missed) and 79.9% precision (97 unnecessary contacts among Normal patients).
- Use case: Situations with limited staffing or budget where reducing outreach volume is important and a small number of missed cases is acceptable.

Scenario 2 - Balanced approach (**recommended**)

- Threshold: **0.6**
- Strategy: Maintain full coverage of pneumonia cases while reducing unnecessary contacts as much as possible within the 100%-recall region.
- Outcome: 100.0% recall (390 out of 390 pneumonia cases correctly identified, 0 missed) and 76.5% precision, with 120 unnecessary contacts and 114 Normal patients correctly not contacted.
- Use case: Standard operation for the retrospective screening program, aligning with the clinical goal of not missing pneumonia patients while keeping outreach workload manageable.

Scenario 3 - Aggressive outreach (maximum redundancy against missed cases)

- Threshold: **0.3**
- Strategy: Use a lower threshold to flag more patients, accepting additional unnecessary contacts while still maintaining 100% recall on the test set.
- Outcome: 100.0% recall (390 out of 390 pneumonia cases correctly identified, 0 missed) and 72.2% precision, with 150 unnecessary contacts among Normal patients. Compared with the recommended threshold of 0.6, this corresponds to 30 additional healthy patients being contacted.
- Use case: Periods of elevated concern (e.g., high-risk season or suspected data drift) where the hospital prefers to err further on the side of contacting more patients rather than risk overlooking potentially vulnerable cases.

5.4 Summary

In this project, generalization is understood as the model's ability to maintain very high recall (few false negatives) on unseen data, even if this comes at the cost of additional false positives, reflecting the clinical priority of not missing pneumonia cases.

On the held-out test set from the same retrospective screening population, the model achieves 100% recall across a wide range of thresholds (0.1 to 0.6). Within this full-recall region, threshold tuning allows the hospital to choose the level of outreach workload it is willing to accept. The recommended operating point at threshold 0.6 achieves 100% recall with the lowest number of false positives in this range, providing a strong balance between clinical safety and operational efficiency.

Overall, the model trained with a focus on recall provides both high performance and operational flexibility. The hospital can make informed, data-driven decisions by adjusting the classification threshold to align with clinical priorities, resource constraints and evolving policies for preventive outreach. Future work could explore data augmentation and transfer learning to potentially improve performance further while preserving a recall-focused behavior.