Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas
Inteligencia Artifical | Grupo: 031

Computational Vision Model for Multi-Label Classification for Chest X-Ray Dataset - DenseNet

May 15, 2025

José Manuel Enríquez Rodríguez
manuel.enriquezr@uanl.edu.mx

Carlos Eduardo García Herrera
eduardo.garciahr@uanl.edu.mx

Martin Alexis Martínez Andrade
martin.martineza@uanl.edu.mx

Daniel Rojas Villareal
daniel.rojasvl@uanl.edu.mx

Diego Adrián Moreno Duarte
diego.morenod@uanl.edu.mx

Abstract

This report describes the development and evaluation of a deep learning model aimed at multi-label classification of medical images, specifically chest X-rays from the NIH Chest X-Ray dataset. For feature extraction, DenseNet121 is used—a deep convolutional neural network architecture that has shown high performance in computer vision tasks—serving as the backbone of the model.
The proposed architecture is composed of two main modules: a Feature Pyramid Network (FPN), responsible for capturing multi-scale information to enhance the spatial representation of the extracted features; and a custom Classification Wrapper, designed to adapt the outputs of the FPN to the label space corresponding to the pathologies present in the dataset.
The report details each stage of the workflow, including input data preprocessing, the configuration of the training environment in PyTorch, the selection of appropriate evaluation metrics for multi-label classification problems, and the analysis of the obtained results. It also discusses the architectural decisions made, the hyperparameters used, and the most relevant observations derived from the model's performance during training and validation.
The results obtained demonstrate the effectiveness of the proposed architecture in addressing the problem of simultaneous detection of multiple medical conditions in radiographic images, highlighting the potential of the approach for future application in AI-assisted clinical settings.

Key words: Deep Learning, Multi-modal Learning, Feature Pyramid Network (FPN), DenseNet121, Tabular and Image Fusion, Medical Image Classification, Multi-label Classification, Neural Network, Convolutional Neural Network (CNN), Feature Extraction, Late Fusion, Adaptive Pooling, PyTorch, Transfer Learning, Hybrid Model, Clinical Decision Support.

Introduction

In recent decades, the healthcare field has significantly benefited from advances in computing and artificial intelligence. From advanced medical image visualization to the use of algorithms capable of assisting in complex diagnoses, technology has transformed the way medical care is approached. In particular, artificial intelligence (AI) applied to computer vision has shown enormous potential in the automatic detection of abnormalities in medical images, helping healthcare professionals improve the accuracy and efficiency of their evaluations.

One of the critical areas in medical diagnosis is the interpretation of chest X-rays, which allows for the detection of a wide range of conditions such as pneumonia, pulmonary edema, cardiomegaly, nodules, among others. Due to the large volume of X-rays that radiologists must review, there is a growing need to develop automated systems that can assist in this task quickly and reliably.

In this context, convolutional neural networks (CNNs) have been widely used for medical image classification tasks. Among them, DenseNet121 stands out as a robust and efficient backbone, capable of capturing deep representations thanks to its dense architecture, where each layer receives as input the outputs of all preceding layers. This feature allows better feature and gradient propagation, improving the model's efficiency in complex classification tasks.

This project proposes a model based on DenseNet121 for the multi-label classification of thoracic diseases from chest X-rays in the NIH Chest X-Ray dataset. The architecture is complemented with a Feature Pyramid Network (FPN) to capture multi-scale information and a custom Classification Wrapper to adapt the model's output to the multi-label structure of the problem. Unlike object detection models such as YOLO, which identify and localize specific objects in an image, this approach focuses on simultaneously predicting the presence of multiple pathologies in a single X-ray.

This report details the model design, data preprocessing process, training procedure, and performance evaluation for the classification task. Additionally, it analyzes the effectiveness of the proposed model and its potential applicability in AI-assisted clinical environments.

1 Hypothesis

1.1 Basis

This project represents a significant contribution to the healthcare sector, especially in the area of medical image diagnosis. The incorporation of artificial intelligence models into clinical practice has the potential to transform how healthcare professionals analyze and interpret X-rays by providing a support tool that optimizes both accuracy and efficiency in pathology detection

Firstly, the use of a multi-label classification model based on advanced computer vision techniques, such as DenseNet121 and Feature Pyramid Network, allows the identification of multiple abnormalities in a single chest image. This capability is especially useful in a clinical setting where a patient may present several conditions simultaneously. Thus, the model not only acts as a preliminary filter highlighting possible areas of interest but also reduces the risk of disease omission by medical staff due to fatigue or workload overload.

Secondly, the proposed system can serve as a complementary tool for experienced physicians, facilitating the prioritization of urgent cases and assisting in diagnosis validation. Its ability to provide fast and reliable predictions improves workflow in medical institutions, allowing more time to be devoted to direct patient care. Additionally, in contexts where the number of specialists is limited, such as rural hospitals or regions with low medical coverage, this technology can be key to ensuring more timely diagnoses.

On the other hand, the model also has educational value. For medical trainees or those in their early years of professional practice, having an AI-based tool can significantly enhance their learning process. By comparing their own observations with the model's predictions, students can develop a better understanding of the visual patterns associated with different diseases, thereby strengthening their diagnostic skills with assisted support.

Finally, the implementation of such solutions helps lay the foundation for a more automated, accessible, and resilient healthcare ecosystem. The combination of medical expertise and artificial intelligence enables progress toward more precise, preventive, and personalized medicine, where response time and diagnostic quality benefit from the strategic use of emerging technologies

Likewise, for us as Computer Science students, it is of great help to implement the knowledge gained throughout the Artificial Intelligence course.

1.2 Expected Results

The main objective of this project is to develop and train a multi-label classification model using DenseNet121 as the primary backbone for feature extraction, aimed at detecting multiple thoracic pathologies from chest X-rays in the NIH Chest X-Ray dataset. To enhance the model's representation capability, a Feature Pyramid Network (FPN) has been incorporated, allowing the integration of information at different scales, along with a custom Classification Wrapper that adapts the model's outputs to the multi-label nature of the problem.

This approach aims to achieve competitive performance on metrics relevant to medical classification, such as precision, recall, F1-score, and Area Under the ROC Curve (AUC) for each label. The model is expected to generalize adequately on unseen images, maintaining a balance between sensitivity and specificity, which is crucial in clinical applications where false negatives can have significant consequences.

In addition to technical performance, one of the key objectives is to ensure that the resulting model is easily reusable and extensible by other researchers, healthcare professionals, or students interested in enhancing their capabilities. For this reason, the project has been structured with a modular and reproducible approach, facilitating hyperparameter tuning, integration of new datasets, or incorporation of architectural improvements.

As a result of this work, it is expected to obtain a functional model that not only serves as a proof of concept for automated thoracic disease classification but also as a solid foundation for future research. By providing an open and well-documented implementation, the model can be reused and refined in various clinical or educational contexts, adapting to different populations or specific needs.

2 Methods

Below is the entire methodology followed throughout the project, specifying the development and research process, as well as the justification for each decision made in the model's development.

2.1 Dataset: NIH-Chest-X-Ray

The dataset was obtained directly from the Hugging Face repository. Below, relevant information about the dataset is described.

2.1.1 Dataset Description

The ChestX-ray8 dataset contains 112,120 frontal chest X-ray images from 30,805 unique patients. Each image is labeled with up to twelve possible thoracic pathologies, automatically extracted from radiology reports using natural language processing techniques. The images can have multiple labels per sample,

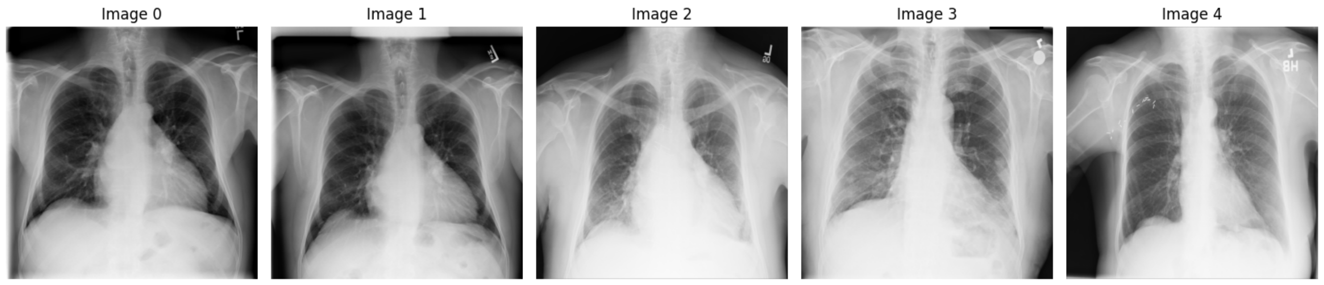


Figure 1: Sample Images Contained in the NIH-Chest-X-ray Dataset

allowing the representation of the simultaneous presence of several diseases in a single X-ray.

2.1.2 Content

- 112,120 frontal chest X-ray images in PNG format with a resolution of 1024×1024 (inside the images folder).
- Metadata for all images (Data_Entry_2017.csv): image index, finding labels, follow-up number, patient ID, patient age, patient sex, view position, original image size, and original pixel spacing.
- Bounding boxes for approximately 1,000 images (BBox_List_2017.csv): image index, finding label, bounding box [x, y, width, height]. [x y] are the coordinates of the top-left corner of each box. [width height] represent the width and height of each box.
- Two data partition files (train_val_list.txt and test_list.txt) are available. The ChestX-ray dataset images are split into these two sets at the patient level. All studies of the same patient will appear only in either the training/validation set or the test set, but not in both.

2.1.3 labeling

Each record in the dataset can be classified into the following diseases:

- Atelectasis
- Consolidation
- Infiltration
- Pneumothorax
- Edema
- Emphysema
- Fibrosis
- Effusion
- Pneumonia
- Pleural Thickening

- Cardiomegaly
- Nodule
- Mass
- Hernia
- No Findings

2.1.4 Label's distribution

Below is a circular chart showing the proportion of images labeled with any of the diseases, since, as mentioned earlier, a single X-ray can have multiple diseases labeled.

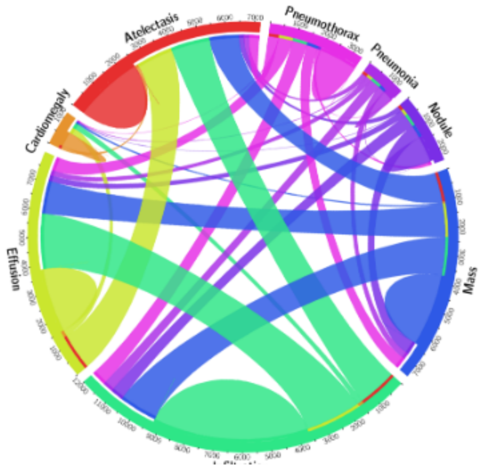


Figure 2: Multi-label Distribution of Diseases in the NIH-Chest-X-ray Dataset

Likewise, a frequency table and a graph is presented for each of the diseases present:

Label	**Count**	**Frequency**	**Label**	**Count**	**Frequency**
No findings	60,361	0.426468	Pleural thickening	3,385	0.023916
Infiltration	19,894	0.140557	Cardiomegaly	2,776	0.019613
Effusion	13,317	0.094088	Emphysema	2,516	0.017776
Atelectasis	11,559	0.081667	Edema	2,303	0.016271
Nodule	6,331	0.044730	Fibrosis	1,686	0.011912
Mass	5,782	0.040851	Pneumonia	1,431	0.010110
Pneumothorax	5,302	0.037460	Hernia	227	0.001603
Consolidation	4,667	0.032973	-	-	-

Table 1: Frequency Table of the Labels Present in the NIH-Chest-X-ray Dataset

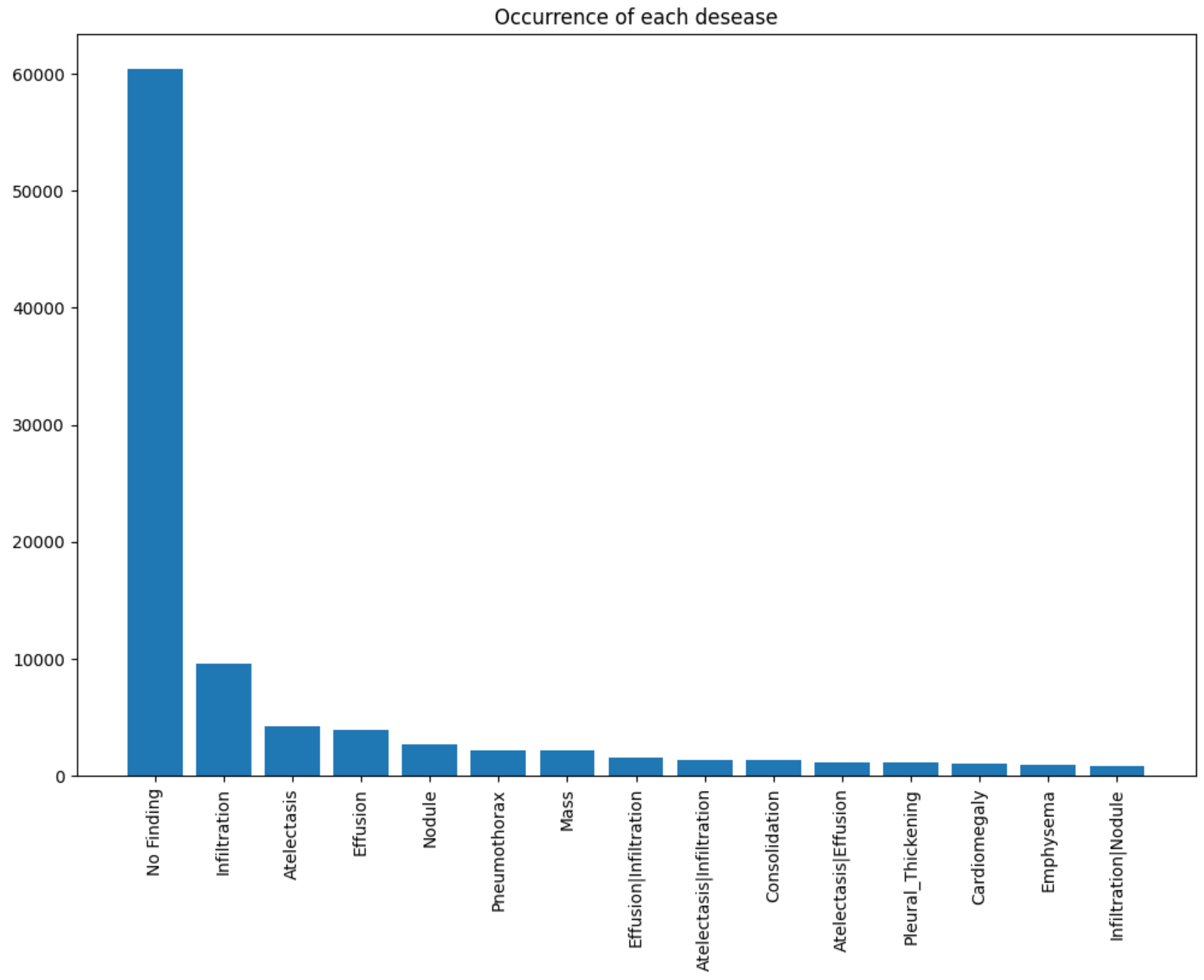


Figure 3: Distribution of Diseases in the NIH-Chest-X-ray Dataset

2.1.5 Patient Information

Graphical representations were generated to summarize demographic information contained in the dataset, specifically the age distribution of the patients and the gender ratio. These visualizations help to better understand the dataset's composition and assess potential biases in the data.

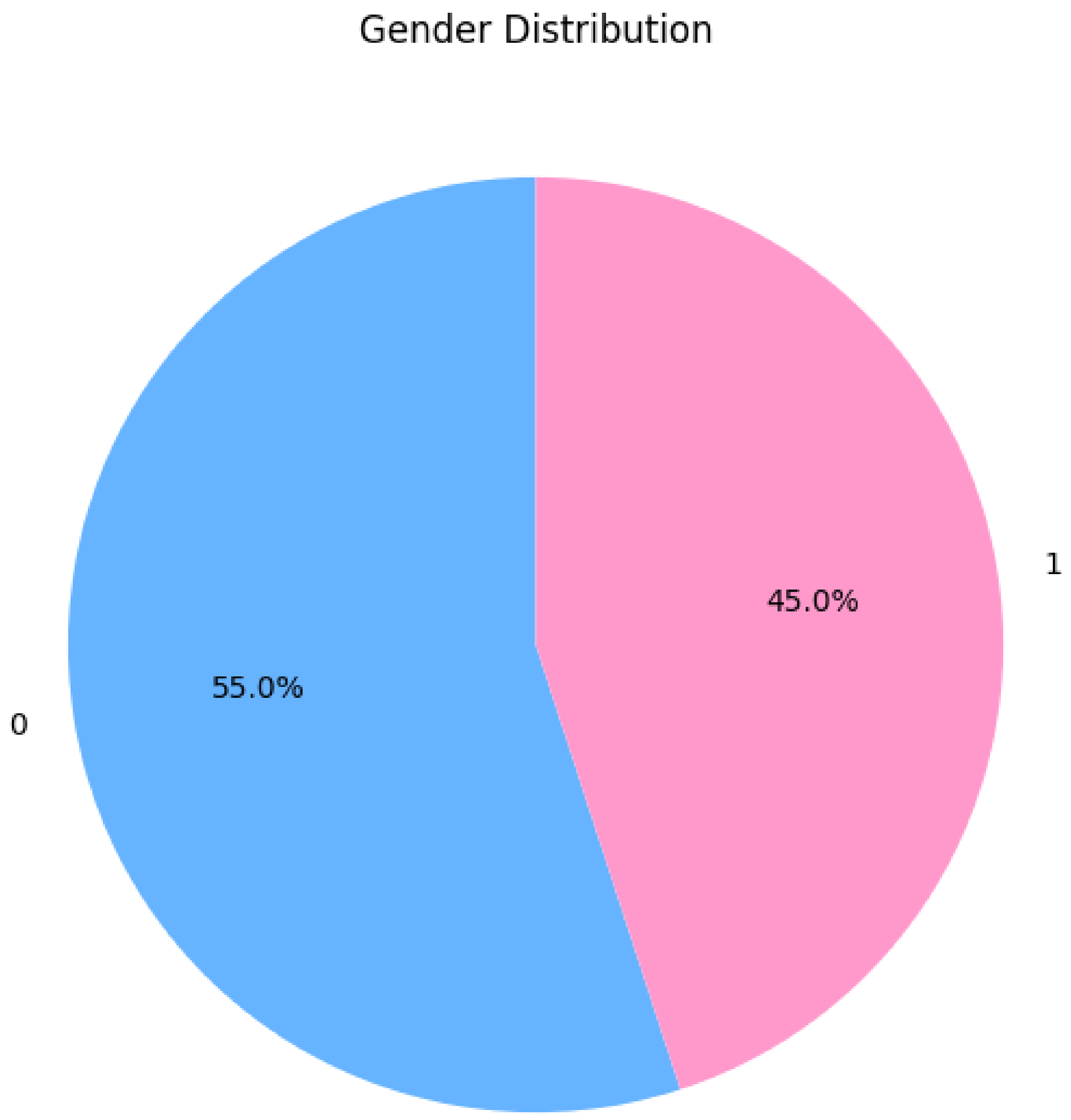


Figure 4: Gender Proportion in the NIH-Chest-X-ray Dataset

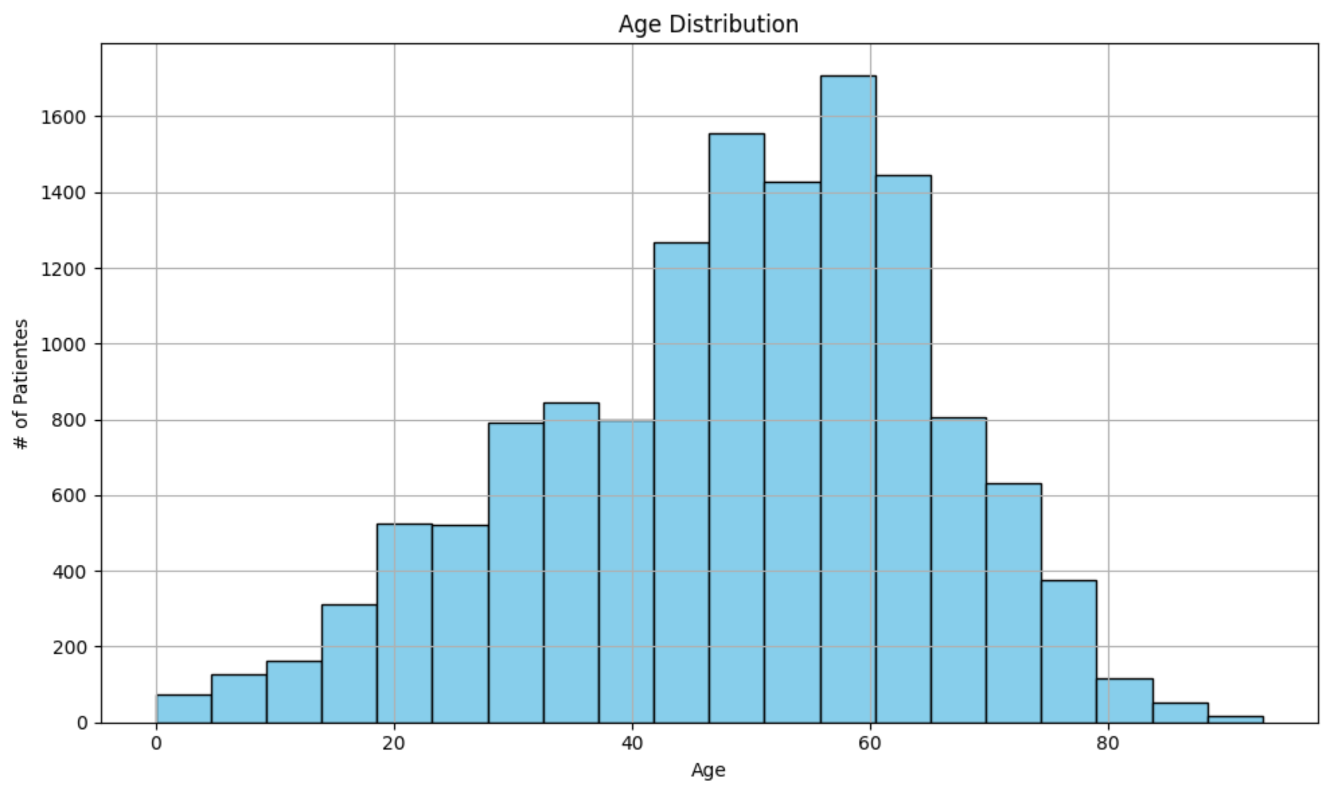


Figure 5: Age Distribution in the NIH-Chest-X-ray Dataset

2.2 Development Environment

The following section describes the development environment used throughout the project, specifying relevant information related to the foundation on which the project was built.

2.2.1 Execution Environment

The selected execution environment was Google Colab, as it offers high-performance resources for training Machine Learning models, both in its free version and in the Pro version. A Google Colab Pro subscription was acquired to access more powerful hardware and benefit from longer execution sessions. The final notebook can be found at the following link, as well as in the Annex section.

Technical Specifications The hardware provided and used in Google Colab has the following specifications:

- CPU: Intel(R) Xeon(R) CPU @ 2.20GHz (6 cores / 12 threads)
- RAM: 53 GB
- GPU: NVIDIA L40 (22.5 GB)
- Storage: 235.7 GB

2.2.2 Machine Learning Framework

As mentioned in previous sections, the PyTorch framework was chosen for the development, creation, and training of the Machine Learning model. Additionally, the libraries used throughout the development process are listed below.

Libraries Used **Data Processing**

- os
- re
- ast
- time
- sleep
- timedelta
- Path
- zipfile
- csv
- glob
- shutil
- json
- numpy
- pandas
- sklearn.utils.resample
- sklearn.model_selection.train_test_split
- sklearn.metrics

Images and Visualization

- cv2
- PIL.Image
- torchvision.transforms.v2
- torchvision.transforms.functional
- matplotlib.pyplot

Machine Learning / Neural Network

- torch
- torch.nn
- torch.nn.functional
- torch.utils.data.Dataset
- torch.utils.data.DataLoader
- torch.utils.data.Subset
- torch.utils.data.random_split
- huggingface_hub.hf_hub_download

2.3 Data processing

The data preprocessing process is explained in the following sections. This preprocessing is very important because it allows us to filter, improve, and enhance the quality of the data our model will receive. The better the data quality, the easier it will be for the model to learn patterns.

2.3.1 One-Hot Encoding of Diseases

For each diagnostic category, 14 new binary columns are created, indicating with a value of 1 the presence of disease N in the corresponding image, or 0 otherwise. In this way, each image is represented by an indicator vector that summarizes its relevant clinical features.

2.3.2 Undersampling

- No findings:

  As evidenced in the Label Distribution section, the dataset shows a significant imbalance, with a higher proportion of patients without pathological findings compared to those with any disease. This imbalance may lead the model to favor predicting healthy cases, compromising its ability to detect relevant clinical conditions. Therefore, it is necessary to apply balancing or restructuring techniques to the dataset to mitigate this bias and improve the model's performance in classifying minority classes. To achieve this, we will retain only 10% of the healthy patients, so the distribution now looks as follows.

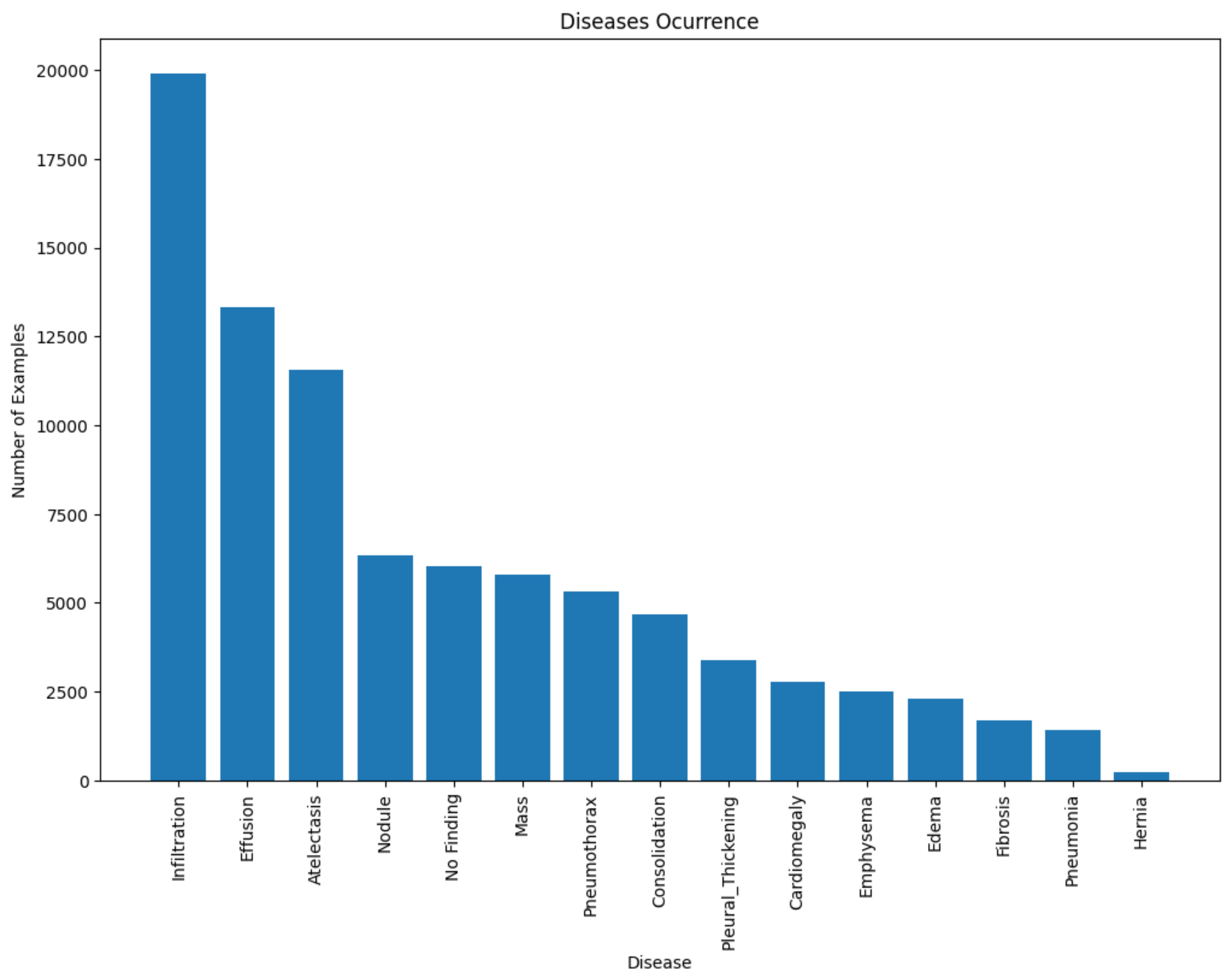


Figure 6: Distribution After Undersampling of No Findings

- Infiltration:

  The same process is carried out for the disease Infiltration, which has a large number of observations. In this case, only 15% of the unique cases of this disease were selected; that is, records where infiltration appeared alongside another disease were not removed.

- Effusion:

  The same process as before was applied; however, in this case, only 50% of the unique cases of Effusion were selected.

- Atelectasis:

  Finally, 50% of the unique cases of Atelectasis were selected.

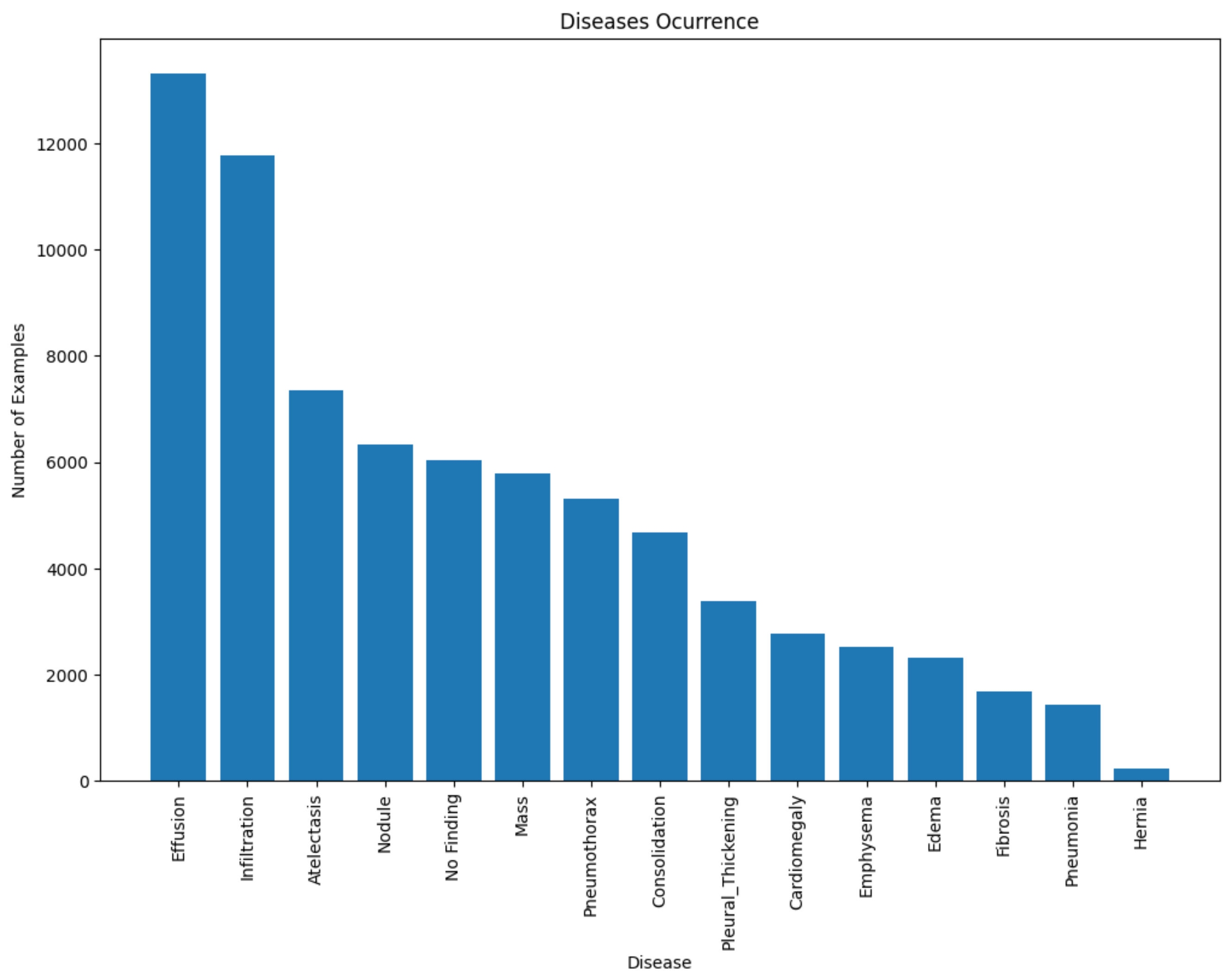


Figure 7: Final Label Distribution in the Dataset

2.3.3 Creation of the disease_vec Column

Subsequently, after obtaining a balanced sample of the dataset, a new column called d̈isease_vecisädded. This column contains, for each image, a binary vector representing the presence (1) or absence (0) of each possible medical condition.

The construction of this vector is carried out from the columns corresponding to the previously generated labels, which contain binary values. These are grouped into an array that allows representing, in a compact and structured way, the set of diseases associated with each image. This format is especially useful for training multi-label classification models, as it allows capturing multiple conditions simultaneously in a single instance.

This column will represent the model's output, also known as Y, since it corresponds to the values to be predicted for each image the model receives.

2.3.4 Patient Information

The model will receive patient data, but the following conversions are necessary:

- Age

  The variable corresponding to patient age is normalized by scaling it to the [0, 1] range to ensure a uniform contribution during model training.

- X-ray Type

  The radiographic view position (View Position) is encoded as follows:

  - 0: Posteroanterior (PA) projections.
  - 1: Anteroposterior (AP) projections.

- Gender

  El sexo se representa de la siguiente manera:

  - 0: Male
  - 1: Female

All this demographic and technical information is subsequently transformed into another vector called "patient_info," which will be used as input data for the model.

2.3.5 Dataset Split

At this stage, the balanced dataset is divided into three subsets:

- Training set (train_df): 72% of the complete dataset
- Validation set (validation_df): 8% of the complete dataset (equivalent to 10% of the training set)
- Test set (test_df): 20% of the complete dataset

To ensure that the label distribution remains consistent across each partition, the stratify option is used, which groups samples based on the initial letters of the diagnostic labels. This guarantees that class proportions are not significantly altered in the test subset, which is crucial for obtaining a fair evaluation of the model's performance.

2.3.6 Preprocessing for train_df

A second cleaning stage was applied exclusively to the training subset (train_df) to avoid any data leakage into the validation or test sets.

First, images corresponding to diseases with fewer than 3,000 occurrences were duplicated using a vertical symmetry transformation. This technique increases the representation of these classes without introducing new external instances.

Next, for diseases with fewer than 1,500 samples, instances were replicated as needed to reach this threshold, aiming to reduce class imbalance

Finally, since the categories ’No Finding’ and ’Atelectasis’ remained overrepresented, they were undersampled, keeping only 25% of the cases of ’No Finding’ and 50% of those associated with ’Atelectasis’.

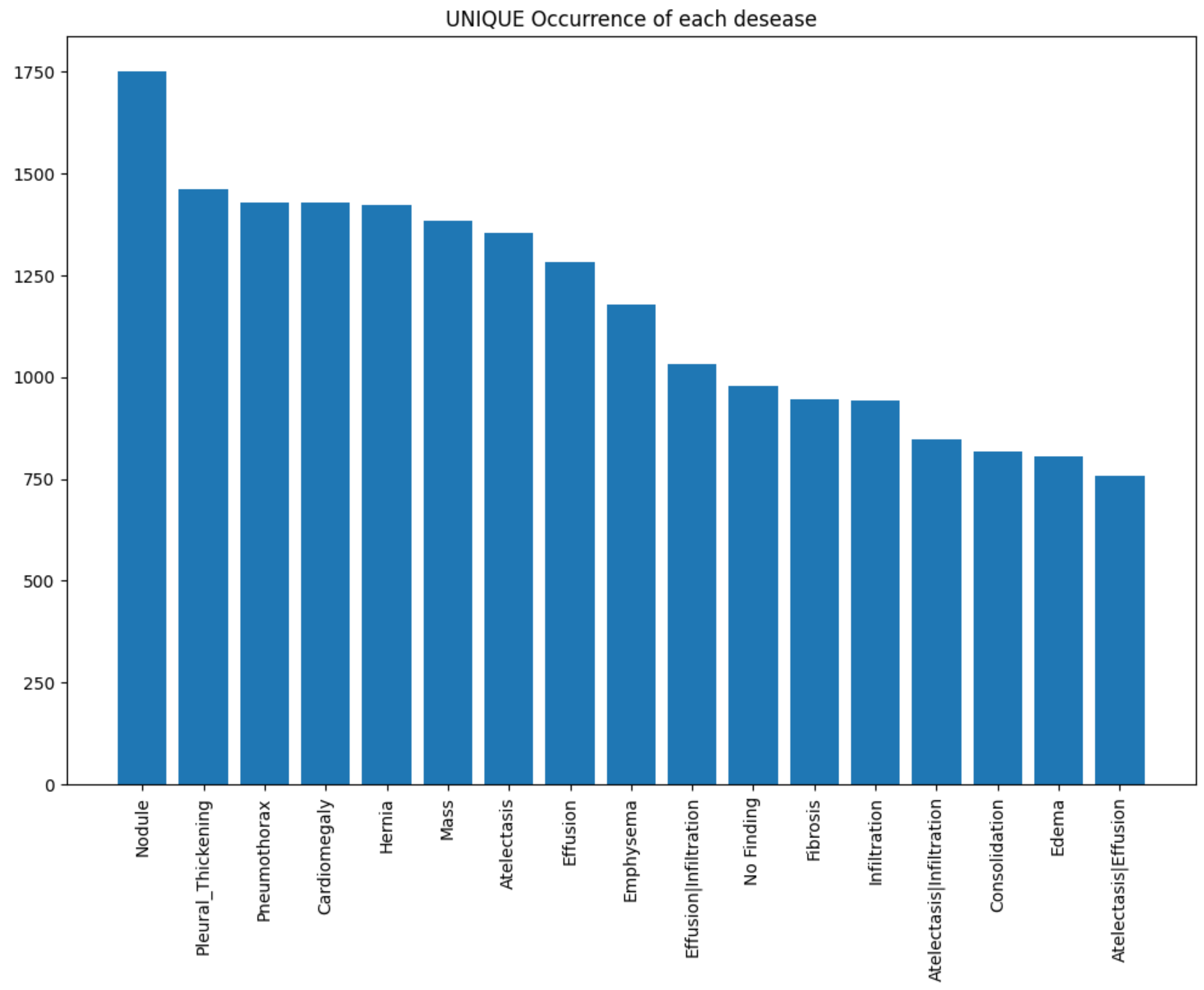


Figure 8: Final Distribution of the Training Dataset

2.3.7 Data Augmentation

In this phase, two DataLoader objects are created, and two sets of transformations are defined to improve the model’s generalization during training, as well as to maintain consistency during validation and testing stages.

- Transformations for Training: train_transform

  The set of transformations applied during training is designed to artificially augment the dataset, introducing variability and robustness in the input data. The applied transformations include:

- Conversion of images to tensors with pixel values normalized between 0 and 1.
- Random cropping of approximately 90% of the original image size, introducing spatial variation.
- Resizing to a fixed size defined by the variable IMAGE_SIZE (e.g., 640×640 pixels), ensuring uniformity in model input.
- Application of random affine transformations, including slight rotations (±7 degrees) and translations (up to 6%) to simulate different acquisition angles.
- Application of Gaussian blur with a kernel size of 3 and sigma in the range [0.2, 1.5], with a 40% probability, to simulate noise in the images.
- Normalization using ImageNet dataset statistics (mean = [0.485, 0.456, 0.406], standard deviation = [0.229, 0.224, 0.225]) to standardize input data.

Transformations for Validation and Testing: val_test_transform

During validation and final evaluation, the focus is on input consistency rather than variability. Therefore, the applied transformations are more conservative:

- Conversion of images to tensors with pixel values scaled between 0 and 1.
- Cropping 90% of the original size to remove edge artifacts, maintaining spatial consistency.
- Resizing to IMAGE_SIZE to match the dimensions expected by the model.
- Normalization using the same ImageNet statistics, ensuring inference conditions are consistent with training.

These transformations allow the model to be trained with enriched data while being evaluated under controlled conditions that represent a real-world environment.

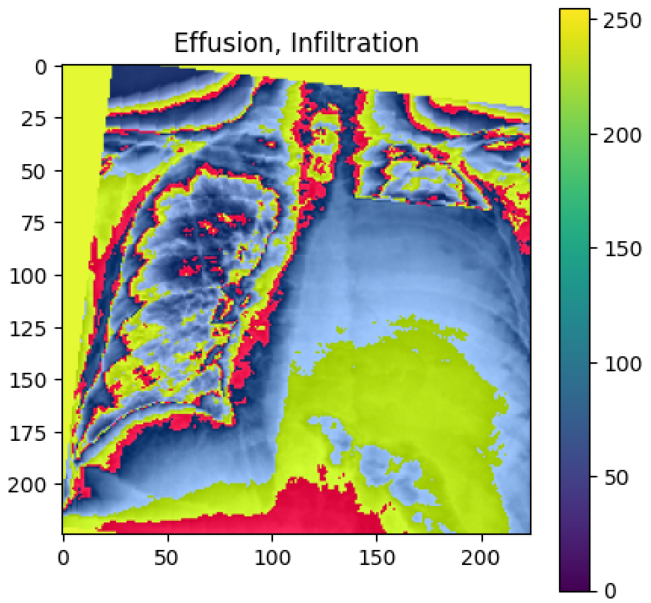


Figure 9: Image After Applying Transformations (Train)

2.4 Neural Network Architecture

The neural network architecture developed is described below. As shown in the following section, it consists of four main blocks: Backbone-FPN, Image Projection, Tabular Data Projection, and Image-Tabular Data Fusion.

2.4.1 Input Data

Medical Image (x_image)

- Type: Tensor
- shape: [B, 3, 224, 224]
- Description: Input image with 3 channels (RGB), sized 224×224 pixels.

Tabular Data (x_tabular)

- Type: Tensor
- shape: [B, 3]
- Description: Numerical vector containing patient information structured per sample.

2.4.2 Output data

Predictions (logits)

- Type: Tensor
- shape: [B, 14]
- Description: Logit values (unactivated) for each of the 14 classes. These values represent the unnormalized evidence the model assigns to each class.

2.4.3 BackBone – DenseNet

Dense Convolutional Networks, or DenseNet, represent a deep neural network architecture designed to improve learning efficiency and feature reuse throughout the network. Unlike traditional architectures where each layer receives input only from the previous layer, in DenseNet each layer is directly connected to all subsequent layers via concatenation. In other words, the input to each layer consists of the outputs of all preceding layers.

This dense connectivity approach offers multiple advantages:

- Improved gradient flow during training, which mitigates the vanishing gradient problem in deep networks.
- Feature reuse, as layers can directly access previous activation maps, reducing redundancy.
- Parameter efficiency, requiring fewer filters and layers to achieve competitive performance.

DenseNet has demonstrated high performance in image classification tasks and has been successfully applied in contexts such as medical analysis, including chest X-rays and other biomedical imaging modalities.

The DenseNet backbone will be responsible for extracting features from the images, due to all the advantages listed above.

The following intermediate layers were extracted from DenseNet:

- features.denseblock2 $\rightarrow$ c3 (512 channels)
- features.denseblock3 $\rightarrow$ c4 (512 channels)
- features.denseblock4 $\rightarrow$ c5 (1024 channels)

These layers will be used and trained through the FPN (Feature Pyramid Network).

2.4.4 FPN - Feature Pyramid Network

It was decided to implement a Feature Pyramid Network (FPN) in the model architecture because there is positive evidence supporting the use of this type of network for feature extraction tasks.

For context, an FPN provides a top-down pathway to build higher-resolution layers from a semantically rich layer.

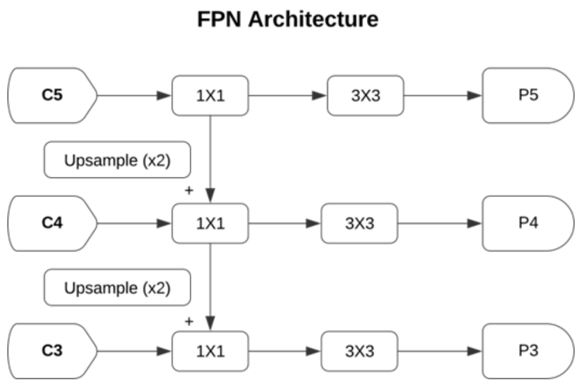


Figure 10: FPN Architecture

As can be seen, the FPN receives as inputs the C5, C4, and C3 layers from DenseNet and then applies a 1x1 convolution using the Conv2d function to each of the channels. This conversion is done to unify the number of channels in each layer so they can be summed together.

Next, for the lateral channels, an upsample (x2) is applied using the function ‘F.interpolate‘ to match the spatial resolution between layers. This function doubles the spatial size, as the FPN is built in a top-down manner—from the deeper layers (more semantic but lower resolution, like C5) to the shallower layers (higher resolution but less semantic, like C3).

Finally, a 3x3 convolution is applied to each of the FPN outputs to remove artifacts created by the upsampling and to homogenize the output.

2.4.5 Classification Wrapper

This part of the neural network combines the two previous components and adds three new components.

The process this neural network performs is as follows:

- BackBone CNN
  - It uses DenseNet121 pretrained on ImageNet.
  - Extracts 3 hierarchical levels (c3, c4, c5) from the backbone.
  - These 3 levels are fed into the FPN to generate a feature pyramid.
- FPN
  - The FPN receives c3, c4, and c5 and converts them into three feature maps, each with 256 channels.
  - It integrates low-, mid-, and high-level features using top-down and lateral connections.
- FPN Output
  - Each FPN output (P3, P4, P5) is reduced to 1×1 using AdaptiveAvgPool2d.
  - Then, they are concatenated into a single vector of size $256 \times 3 = 768$ per sample.
- Image projection
  - It converts the 768-dimensional features to 256 using a dense (fully connected) layer followed by a ReLU activation.
- Tabular information
  - It converts the tabular data (e.g., age, gender, medical history) into an 8-dimensional vector.
  - Then, it applies a ReLU activation function to help learn complex relationships within the patient data.
- Combination of Tabular Information with Image
  - It combines the image and tabular data.
  - Passes them through dense layers and applies Dropout for regularization.
  - The final output has num_classes = 14 (each class represents a disease).

Below is the complete architecture of the trained neural network:

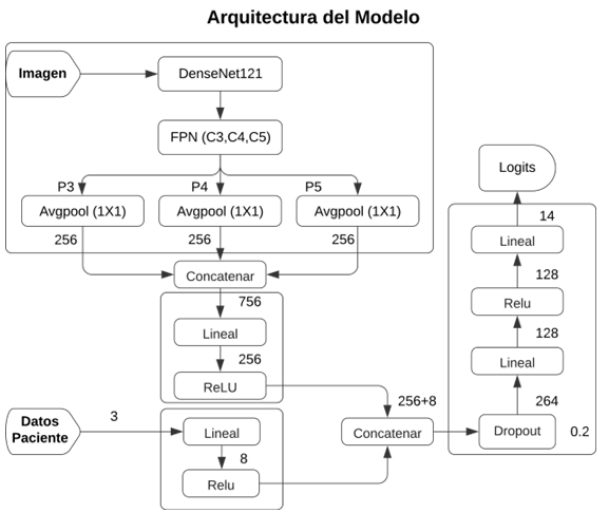


Figure 11: Model Architecture

As shown in the previous figure, the model is composed of 4 main blocks. You can find the code that builds this neural network in the Google Colab Notebook.

2.5 Training Process

As mentioned in the Development Environment section, the resources available to us were very helpful for training the model multiple times with different parameters and configurations (all these versions can be seen at the following link). The final model was trained with the following parameters:

2.5.1 Training Parameters

- BatchSize

  A batch size of 160 was set, which fully utilized the VRAM of the Nvidia L4 GPU. If you want to train the model in an environment with lower specifications, it is recommended to reduce the batch size. For the T4 GPU, we tested with a batch size of 100.

- Image Resolution

  The image resolution was set to 224x224 pixels, as it represents a balance between quality and performance, and DenseNet was originally trained at this resolution.

  Tests were also conducted with resolutions of 580, 512, and 480 pixels, but the training process was drastically affected, and the feature extractor's performance was negatively impacted.

- Optimization Criterion

  Binary Cross Entropy with Logits Loss (nn.BCEWithLogitsLoss) was used, as it is a function that combines:

  - Sigmoid activation.

– Binary Cross Entropy Loss.

This function is used for multi-label classification problems.

Custom weights for each of the classes in the dataset were added to this function as an argument, in order to counteract the existing imbalance. The calculation of the weights is described below.

- Custom loss weights
According to the number of observations in the training dataset for each of the diseases, a custom weight is calculated: a higher weight is assigned to classes with fewer positive examples (less frequent), and a lower weight to the more common classes. These weights force the model to learn more from the less frequent classes. The resulting weights are as follows:

Disease	**Weight**
Hernia	3.0943
Nodule	2.1768
Fibrosis	3.0209
Edema	2.8127
Cardiomegaly	2.5823
Effusion	1.6668
Pneumothorax	2.3087
Pleural Thickening	2.4494
Consolidation	2.4346
Emphysema	2.6907
Pneumonia	3.0943
Mass	2.2459
Atelectasis	1.8179
Infiltration	1.6402

Table 2: Custom weights for the loss function

It is important to highlight that the model underwent 3 training cycles. In the first 2 cycles, the weights from the previous table were used, but for the third cycle, the weights were manually set for the following diseases because the model was not correctly identifying them.

Disease	**Weight**
Pneumonia	4.5
Fibrosis	3.8
Consolidation	4.3

Table 3: Manual weights for the weak classes

- Learning Rate

A learning rate of 0.001 was set along with a scheduler, which is specified below.

Scheduler: It monitors the validation loss (val_loss) metric during training. If this metric does not improve for 2 epochs, the learning rate is reduced by half.

- Epochs

  Three training cycles were performed, each consisting of 5 epochs.

2.5.2 Train Cycle

During the training cycle, the training loss value was recorded, as well as the time for each step throughout the epoch.

On average, an epoch consisted of 180 steps with a duration of 25 minutes per training epoch. The entire training process lasted 280 minutes (4 hours and 40 minutes).

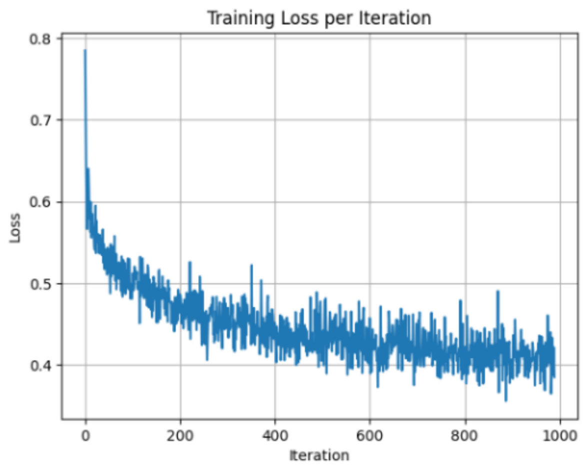


Figure 12: Training Loss per Iteration (Epochs 10-15)

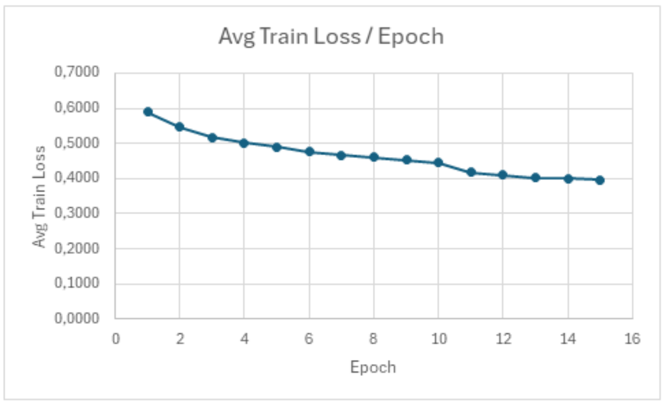


Figure 13: Training loss per epoch

2.5.3 Validation cycle

The validation cycle is very helpful to understand how well the model is learning. Below are the main points covered in this cycle.

- Our model outputs logits (raw data representing the presence of each class). Before making predictions with these data, it is necessary to apply the torch.sigmoid function, which outputs the probabilities of each class being present. However, due to the imbalance in the dataset, it is recommended to use a dynamic threshold for each class (instead of a fixed probability for all) to decide whether the disease is present according to its customized threshold. We conducted tests calculating these thresholds and have three functions that compute the thresholds:

  - find_optimal_thresholds_by_roc

    This function searches for the optimal threshold for each class using the ROC curve, aiming to maximize Youden's J statistic, defined as sensitivity minus (1 - specificity). In other words, it finds the point on the ROC curve that provides the best balance between the true positive rate and the false positive rate. For each class, the ROC curve is calculated using sklearn.metrics.roc_curve, the J value is computed for each possible threshold, and the threshold that maximizes J is chosen. In cases where a class is absent or fully present (i.e., only zeros or ones in the labels), it returns a default threshold of 0.5. This function is useful when adjusting thresholds based on the model's ROC characteristics, without directly prioritizing precision or recall.

  - find_optimal_thresholds_with_precision_constraint

    This function finds the best threshold per class based on the maximum F1-score, provided that a minimum precision set by the user is met. It iterates through a set of possible thresholds (by default, from 0.1 to 0.85 in increments of 0.05) and, for each class, discards those that do not meet the minimum required precision (default is 0.30). From the valid thresholds, it selects the one that produces the highest F1-score. This function is especially useful in contexts where strict control over false positives is a priority, such as in medical diagnostics, where an incorrect prediction can have significant consequences.

  - find_optimal_thresholds_with_dynamic_adjustment

    This variant also aims to maximize the F1-score under a minimum precision constraint but adds a dynamic adjustment if the performance (measured by F1-score) is insufficient. If, after applying the previous criterion, the best F1-score is still below a defined threshold (default 0.30), the function lowers the threshold further (according to an adjustment factor, default 0.05) to allow for higher recall. This can be crucial in cases where detecting more positives is necessary, even if it means increasing false positives.

    The function that gave us the best results was number 3.

    These dynamic thresholds are stored in a dictionary structure within a JSON file.

- Validation loss

The validation loss value during training is shown in the following graph:

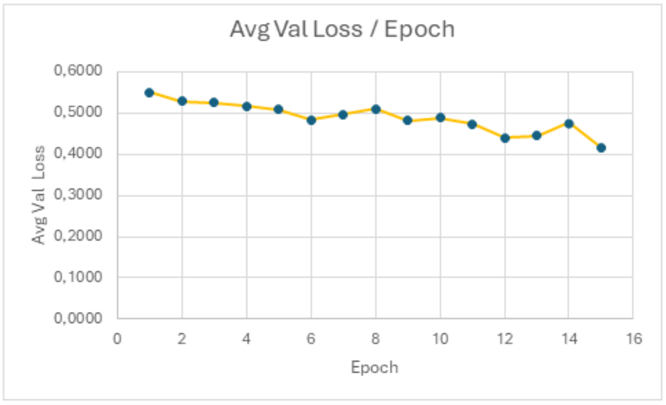


Figure 14: Validation loss per epoch

As can be seen in the graph, the model managed to reduce its validation loss value, but it has plateaus across several epochs and sometimes worsened between epochs.

- Accuracy (mean)

The accuracy value obtained during training is shown in the following graph:

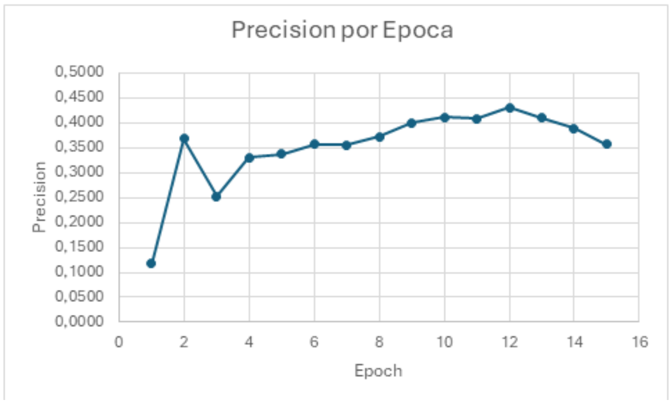


Figure 15: Accuracy per epoch

As can be seen, after epoch 12, the model began to decrease its accuracy.

- Recall (mean)

The recall value obtained during training is shown in the following graph:

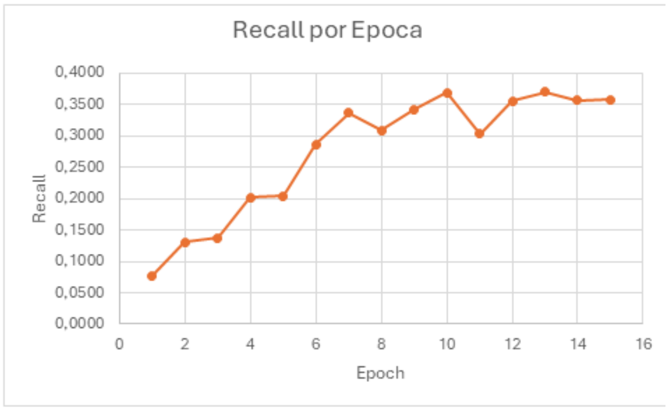


Figure 16: Recall per epoch

Unlike accuracy, the recall remained constant after epoch 12.

- Macro F1 (Mean)

The Macro F1 value obtained during training is shown in the following graph:

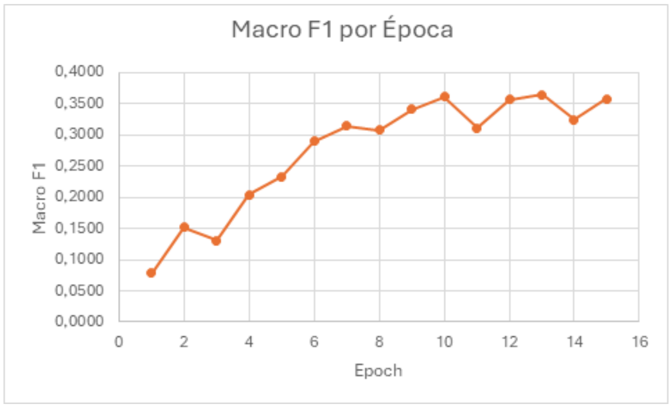


Figure 17: Macro F1 per epoch

The Macro F1 Score did not show significant improvement after epoch 10.

- AUC (Mean)

The AUC value obtained during training is shown in the following graph:

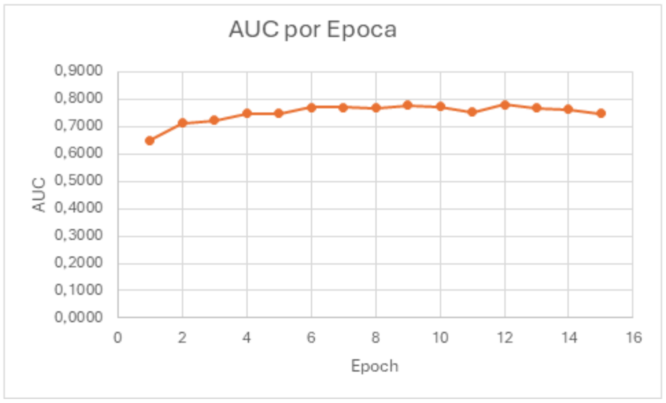
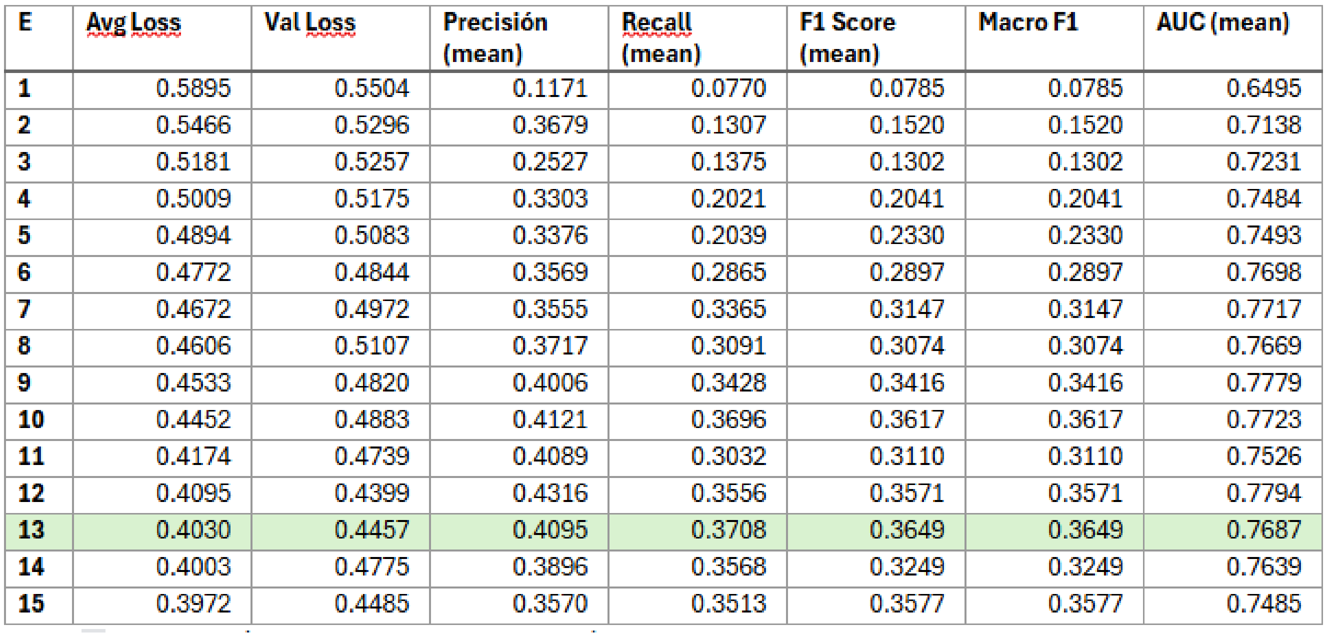


Figure 18: AUC per epoch

The AUC value remained constant throughout the training process, with the best value obtained at epoch 12.

All these data can be seen in the following table:



E	Avg Loss	Val Loss	Precisión (mean)	Recall (mean)	F1 Score (mean)	Macro F1	AUC (mean)
1	0.5895	0.5504	0.1171	0.0770	0.0785	0.0785	0.6495
2	0.5466	0.5296	0.3679	0.1307	0.1520	0.1520	0.7138
3	0.5181	0.5257	0.2527	0.1375	0.1302	0.1302	0.7231
4	0.5009	0.5175	0.3303	0.2021	0.2041	0.2041	0.7484
5	0.4894	0.5083	0.3376	0.2039	0.2330	0.2330	0.7493
6	0.4772	0.4844	0.3569	0.2865	0.2897	0.2897	0.7698
7	0.4672	0.4972	0.3555	0.3365	0.3147	0.3147	0.7717
8	0.4606	0.5107	0.3717	0.3091	0.3074	0.3074	0.7669
9	0.4533	0.4820	0.4006	0.3428	0.3416	0.3416	0.7779
10	0.4452	0.4883	0.4121	0.3696	0.3617	0.3617	0.7723
11	0.4174	0.4739	0.4089	0.3032	0.3110	0.3110	0.7526
12	0.4095	0.4399	0.4316	0.3556	0.3571	0.3571	0.7794
13	0.4030	0.4457	0.4095	0.3708	0.3649	0.3649	0.7687
14	0.4003	0.4775	0.3896	0.3568	0.3249	0.3249	0.7639
15	0.3972	0.4485	0.3570	0.3513	0.3577	0.3577	0.7485

Figure 19: Table with Training and Validation Metrics

2.5.4 Saving Criterion

After each validation cycle, the best model so far is saved. During training, we tested the following metrics to consider a model as "The best":

- AUC

  It evaluates the model's ability to distinguish between positive and negative classes for each label.

  - Why it matters: In multilabel classification, each class is treated as a binary problem, and AUC

provides a robust measure of the model's discrimination per class, regardless of the decision threshold.
  - Advantage: It is not affected by class imbalance, which is common in multilabel tasks (for example, some diseases are much rarer than others).
  - Useful for assessing how well the model separates positives from negatives, without relying on a specific threshold.

- Macro F1 (Selected)
  - What it measures: It is the average of the F1-score for each class, giving equal weight to every label regardless of how frequent or rare it is.
  - Why it matters: In multilabel classification, rare labels tend to be ignored if a global metric like accuracy or micro F1 is used. Macro F1 forces the model to perform well on all labels equally.
  - Advantage: It favors models that have balanced performance between common and rare classes.
  - Ideal when all classes are equally important.
  - This was the criterion we considered most important for selecting the "Best model," with the model from epoch 13 being chosen. The results shown are based on this model.

- Accuracy
  - What it measures: Of the cases the model predicted as positive, how many were actually positive.
  - Why it matters: In sensitive applications (like medical diagnosis), minimizing false positives is crucial. A model with high precision is conservative, predicting positive only when fairly confident.
  - Disadvantage: This may come at the cost of low recall (the model might miss true positives).
  - Key when false positives are more costly than false negatives.

2.5.5 Resulting Files

When the training and results analysis process finishes, five files are saved:

- validation_metrics_[Model's name].csv

  It contains the model's metrics obtained using the test dataset (test_df).

- configuración.txt

  Contains the configuration and logs of the training process.

- optimizer_classification.pt

  Contains the current state of the optimizer.

- model_classification.pt

  Contains the weights of the model architecture.

- thresholds.json

  Contains the custom thresholds for each disease.

- The files for the final model are located in the following archive: `FINAL_DenseNetTuned.zip`

3 Results

3.1 Acceptance Thresholds

First, the thresholds calculated during the validation process and stored in the thresholds.json file will be analyzed.

Disease	**Weight**
Hernia	0.20
Nodule	0.50
Fibrosis	0.65
Edema	0.75
Cardiomegaly	0.40
Effusion	0.50
Pneumothorax	0.25
Pleural Thickening	0.45
Consolidation	0.50
Emphysema	0.40
Pneumonia	0.45
Mass	0.25
Atelectasis	0.35
Infiltration	0.45

Table 4: Custom Thresholds by Disease

As a recap, function number 3 was used to find the best thresholds, following this procedure:

- It explores thresholds between 0.10 and 0.85, in steps of 0.05.
- For each threshold t, it computes the following:
  - Accuracy
  - F1-score
- If the threshold meets the condition precision 0.40, and the F1-score is the highest found so far, it is saved as a candidate.
- After evaluating all possible thresholds:

– If the best F1-score found is still lower than 0.30, a downward adjustment is applied to the threshold (by 0.05) to improve recall, allowing more true positives to be detected in difficult classes.

That is, we can anticipate the following (at least for the validation dataset):

The threshold values obtained per disease reflect the model's individual performance for each class, based on its ability to balance precision and F1-score. For example, diseases such as Edema (threshold of 0.75) and Fibrosis (0.65) have high thresholds because the model is able to make reliable predictions for these classes, achieving good precision without the need to lower the threshold. In contrast, classes like Pneumothorax and Mass, with thresholds of 0.25, require more permissive decisions. This suggests that the model has difficulty detecting them confidently, so the threshold is reduced to increase recall and prevent positive cases from being overlooked.

The most extreme case is Hernia, with a threshold of 0.20, indicating that the model likely produces many false negatives for this disease if higher thresholds are used. Therefore, the adaptive adjustment is applied to allow any moderately high probability to already count as a positive prediction.

3.2 Validation Metrics

3.2.1 Exact Match Ratio

This metric measures the percentage of times the model predicted exactly all the diseases present in a chest X-ray. It is precisely the strictest metric used to evaluate the accuracy of multi-label prediction models. An accuracy percentage of around 14.21% was obtained, which is a low percentage considering that most images have only one disease to predict. This may indicate that, although the correct disease is predicted, additional incorrect diseases are also being predicted.

3.2.2 Hamming Loss

The Hamming Loss measures the percentage of false positives and false negatives produced by the model for each class, out of the total predictions. This means that the closer the value is to zero, the more accurate the model has been in predicting the value for each class. The result obtained was 0.1326, or basically 13.26% errors across all classes. This implies that, most of the time—around 87% the model correctly predicted the presence or absence of a disease in an image. This can indicate good overall performance, as it shows that the model was not simply predicting all possible values to achieve 100% accuracy.

3.2.3 Recall

There are two types of Recall: micro Recall, calculated by considering the total true positives over all positives across all classes; and macro Recall, calculated by averaging the individual recall of each class. The closer the value is to one, the more true positives and fewer false negatives the model has, meaning higher sensitivity.In this case, a micro recall of 0.43 and a macro recall of 0.38 were obtained, indicating that basically around 40% of the times a disease was present, the model predicted a positive value. This is a poor result, since the model should at least be able to identify when a disease is present in an image, regardless of whether other incorrect diseases were also identified.

3.2.4 Label Ranking Average

This metric measures the percentage of times the highest prediction scores were assigned to all the classes present in an image. In other words, if there are 3 classes in the image, a correct ranking means that those classes, regardless of order, have the highest prediction scores compared to the other classes.In this case, a value of 64% of correctly ranked classes was obtained. This is a good value, as it approaches 70%. However, this does not mean that the ranked classes had the necessary prediction scores to be considered present in the image, so it is not a definitive indicator of the model's overall performance.

3.2.5 Brier Score

The Brier score is used to measure the accuracy of the model by assigning more weight to incorrect predictions, especially those caused by the model's overconfidence (for example, a prediction of 0.99 is penalized more if the true value was 0 than predicting 0.5 for a true value of 0).The following table shows the predictions obtained for all classes:

Class	**Brier Score**
Hernia	0.0046
Nodule	0.1189
Fibrosis	0.0721
Edema	0.0447
Cardiomegaly	0.0397
Effusion	0.1487
Pneumothorax	0.0782
Pleural Thickening	0.1237
Consolidation	0.0922
Emphysema	0.0372
Pneumonia	0.0431
Mass	0.1011
Atelectasis	0.1587
Infiltration	0.1836

Table 5: Brier Score by Class

Since most of the obtained values are below 0.10, this implies that, in general, the difference between the predicted percentage and the true value is not very large. This indicates that the model generally had good accuracy when predicting the diseases.

3.2.6 Global F1 Score

The F1 score evaluates the percentage of true positives with respect to true positives, false negatives, and false positives; it averages across all classes if macro, and over all instances if micro. In this case, it is similar to recall, but includes all false positives in the percentage. For micro F1, a value of 0.41 was obtained, and for macro F1, 0.32. These values are similar to those obtained for recall, but now with the global F1, it is observed that generally about 32% of the time a disease was correctly identified in the image, while the rest of the time the model simply made errors.

3.2.7 F1 Score (Class)

Now the F1 Score is shown below but by class in the following table:

Class	**F1 Score**
Hernia	0.4237
Nodule	0.2261
Fibrosis	0.1538
Edema	0.1548
Cardiomegaly	0.5475
Effusion	0.6020
Pneumothorax	0.4736
Pleural Thickening	0.2565
Consolidation	0.0259
Emphysema	0.4416
Pneumonia	0.0330
Mass	0.3814
Atelectasis	0.4695
Infiltration	0.4147

Table 6: F1 Score by Class

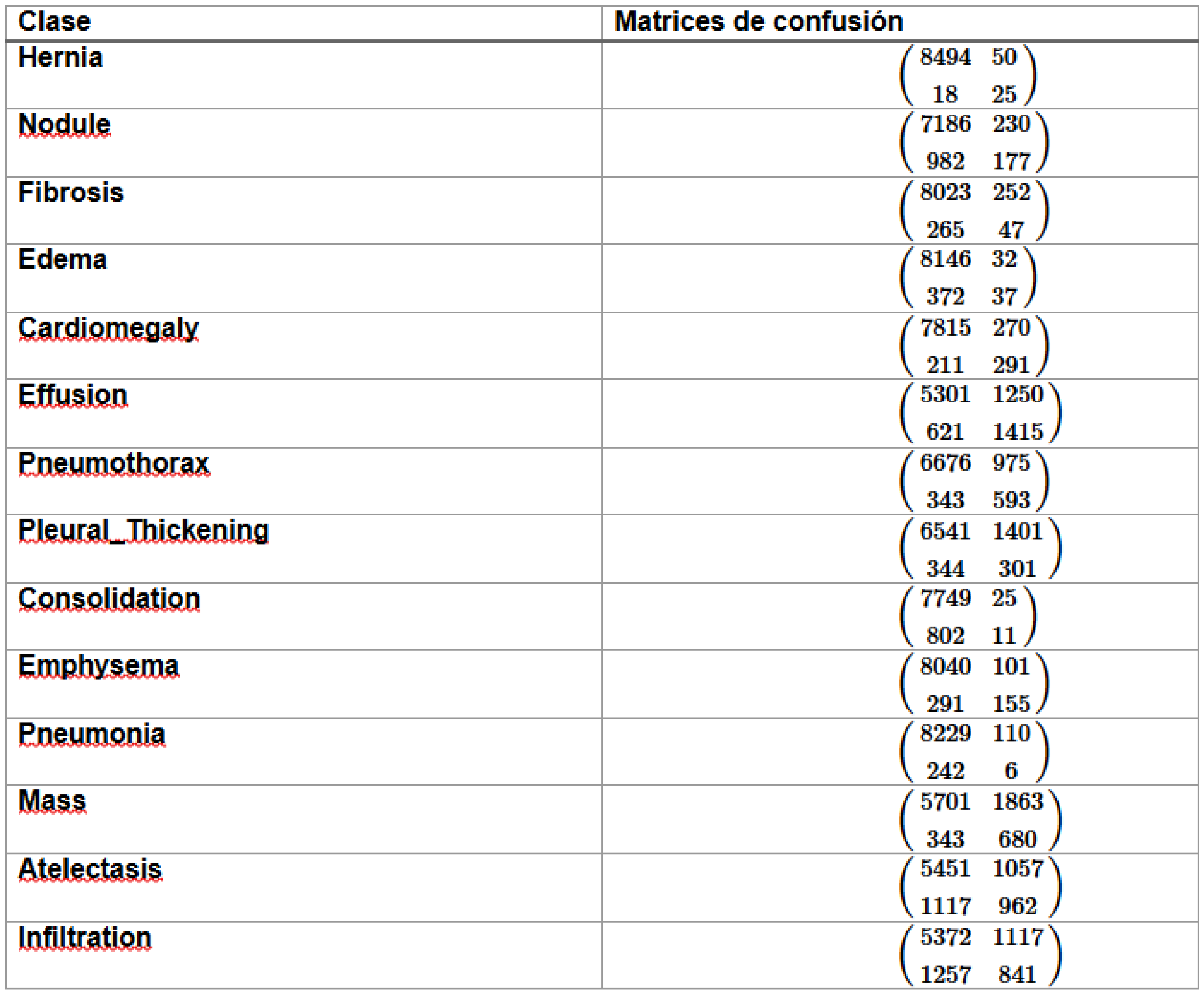
In general, the results hover around 40% and 30% accuracy. However, the table shows some values where the accuracy rate is very low, specifically when identifying diseases such as Pneumonia, Consolidation, Edema, and Fibrosis. This clearly highlights the cases where more training is needed.

3.2.8 Accuracy

This metric averages the accuracy when predicting the value of each class across all images, meaning it checks the proportion of true positives and true negatives. An overall result of 86.74% was obtained, which, although a good indicator of accuracy, does not necessarily mean that the model is predicting correctly. Since true negatives are also taken into account, the accuracy increases considerably compared to the metrics obtained for recall and F1-score, as it is common for a disease to be absent in most images. The accuracy indicator basically confirms that, at least, the model almost always does not predict a disease when it is not present.

3.2.9 Confusion Matrices

Confusion matrices show the number of true positives and true negatives, as well as false positives and false negatives predicted for each class. Below are the confusion matrices for all classes:



Clase	Matrices de confusión
Hernia	$\begin{pmatrix} 8494 & 50 \\ 18 & 25 \end{pmatrix}$
Nodule	$\begin{pmatrix} 7186 & 230 \\ 982 & 177 \end{pmatrix}$
Fibrosis	$\begin{pmatrix} 8023 & 252 \\ 265 & 47 \end{pmatrix}$
Edema	$\begin{pmatrix} 8146 & 32 \\ 372 & 37 \end{pmatrix}$
Cardiomegaly	$\begin{pmatrix} 7815 & 270 \\ 211 & 291 \end{pmatrix}$
Effusion	$\begin{pmatrix} 5301 & 1250 \\ 621 & 1415 \end{pmatrix}$
Pneumothorax	$\begin{pmatrix} 6676 & 975 \\ 343 & 593 \end{pmatrix}$
Pleural_Thickening	$\begin{pmatrix} 6541 & 1401 \\ 344 & 301 \end{pmatrix}$
Consolidation	$\begin{pmatrix} 7749 & 25 \\ 802 & 11 \end{pmatrix}$
Emphysema	$\begin{pmatrix} 8040 & 101 \\ 291 & 155 \end{pmatrix}$
Pneumonia	$\begin{pmatrix} 8229 & 110 \\ 242 & 6 \end{pmatrix}$
Mass	$\begin{pmatrix} 5701 & 1863 \\ 343 & 680 \end{pmatrix}$
Atelectasis	$\begin{pmatrix} 5451 & 1057 \\ 1117 & 962 \end{pmatrix}$
Infiltration	$\begin{pmatrix} 5372 & 1117 \\ 1257 & 841 \end{pmatrix}$

Figure 20: Confusion matrices

3.2.10 ROC Curve

The ROC curve (Receiver Operating Characteristic) plots the values between the false positive rate (on the x-axis) and the true positive rate (on the y-axis), obtained by moving the minimum predicted probability threshold from 0 to 1 to consider a prediction as positive (i.e., that the class is considered present). In a good ROC curve, the curve should tend towards the upper left corner. Below is the graph obtained for the trained model:

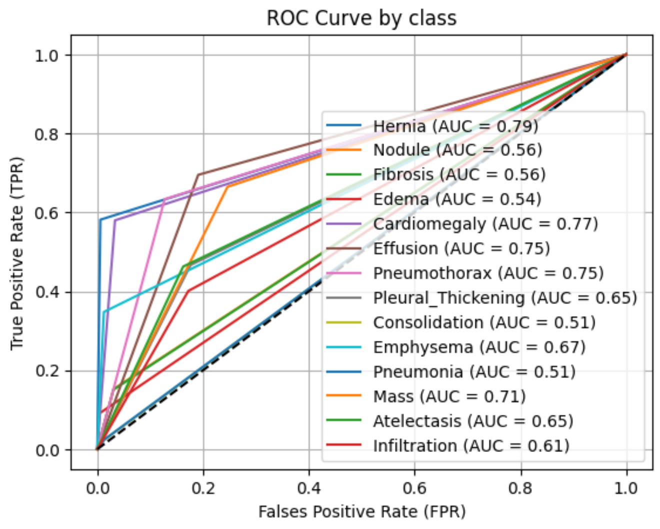


Figure 21: ROC Curve

The graph shows a slight tendency of the data towards the left, demonstrating that at least the model is not making random predictions in most cases. However, it is still observed that the model has room for improvement.

3.3 Examples

Below are some predictions made by the model:

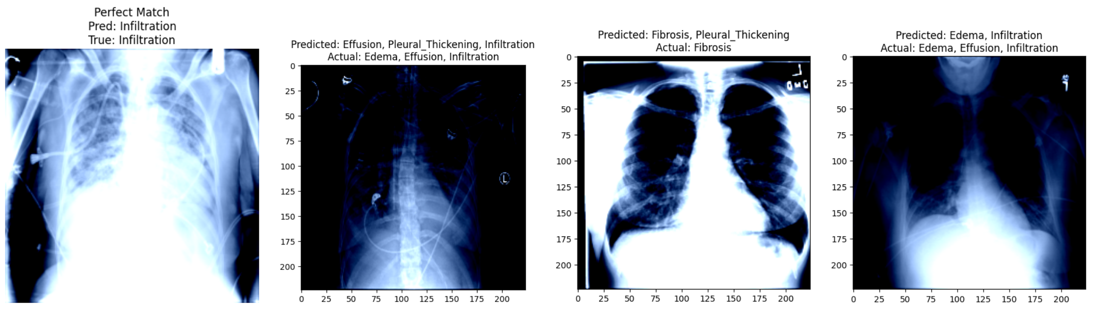

Perfect Match
Pred: Infiltration
True: Infiltration
Predicted: Effusion, Pleural_Thickening, Infiltration
Actual: Edema, Effusion, Infiltration
Predicted: Fibrosis, Pleural_Thickening
Actual: Fibrosis
Predicted: Edema, Infiltration
Actual: Edema, Effusion, Infiltration

Figure 22: prediction examples

As can be seen, the model correctly predicts some diseases, but it also presents some false positives as well as false negatives. Our model has significant areas of opportunity and improvement.

4 Conclusion

From the analysis of the obtained results, the following relevant observations can be drawn regarding the behavior of the multilabel classification model on the test set:

- Adequate performance on frequent classes

The model showed solid performance on classes with a higher number of samples, such as Effusion (F1-score = 0.60), Atelectasis (F1-score = 0.47), and Infiltration (F1-score = 0.41). This behavior indicates that the model is able to learn effective representations when it has access to a sufficient number of examples during training.

- Acceptable results in complex and infrequent classes

  Despite their low frequency in the dataset, some classes such as Hernia (F1-score = 0.42) and Pneumothorax (F1-score = 0.47) achieved reasonably satisfactory metrics. This suggests that the model is capable of detecting useful patterns even in less represented clinical conditions.

- Low performance in specific classes

  The classes Pneumonia (F1-score = 0.03) and Consolidation (F1-score = 0.03) reflect poor performance. These conditions may exhibit less distinctive visual features or overlap with other pathologies, making automatic detection more difficult. Additionally, the low number of examples might be limiting the model's ability to generalize for these labels.

- Impact of Class Imbalance

  A significant negative impact of the imbalance in the dataset was observed, particularly in classes such as Fibrosis (F1-score = 0.15) and Pleural_Thickening (F1-score = 0.26), which have lower representation. This highlights the need to apply mitigation techniques such as class reweighting, oversampling, or synthetic data generation (e.g., SMOTE).

- Mismatch between precision and recall

  The model showed a tendency to favor recall (sensitivity) in some classes (Mass, Hernia) and precision in others (Edema, Emphysema), which could be adjusted according to clinical requirements. For example, in medical contexts, it is common to prefer models with high sensitivity to minimize false negatives.

- Importance of class-specific threshold adjustment

  The results obtained reinforce the importance of using adaptive thresholds per class to convert probabilities into binary predictions, as was done in this work. This strategy is particularly useful in multi-label contexts with high class imbalance, allowing for improved F1-score and other relevant metrics for each label.

Appendix A: Resources and Dataset Sources

This appendix provides an overview of the datasets and development resources used throughout this project.

A.1 Datasets

- **NIH Chest X-ray Dataset** — Available on Hugging Face:
  `https://huggingface.co/datasets/alkzar90/NIH-Chest-X-ray-dataset`
  This dataset contains more than 100,000 frontal-view X-ray images from over 30,000 patients, labeled with up to 14 thoracic disease categories.

- **ChestXRay15GB** — Hosted by Carlos56g on Hugging Face:
  `https://huggingface.co/datasets/Carlos56g/ChestXRay15GB`
  A 15 GB collection of chest X-ray images structured to support deep learning tasks, with multilabel annotations.

- **Additional Dataset** — Provided via OneDrive:
  `https://1drv.ms/f/c/4b0a54b2a0ed7ea3/Euz8-BKzHhJPtANEs3hI0J4B4R4uAeSF8PN46utVUtBD6w?e=nAooCo`
  This dataset includes supplementary chest X-ray images and metadata used for validation and experimentation purposes.

A.2 Code and Implementation Resources

- **Colab Notebook: Model Development and Training**
  `https://colab.research.google.com/drive/1kiE2kPVV9gReXCU7kyj7lowVFoQfZtNp?usp=sharing`
  This Google Colab notebook contains all the training, preprocessing, and model evaluation steps. It serves as the main experimentation environment.

- **GitHub Repository: IAPIA-DanielRojas**
  `https://github.com/DanielRojas1920/IAPIA-DanielRojas`
  The source code, scripts, and additional documentation are hosted here. This repository includes training utilities, configuration files, and model checkpoints.

All resources were used exclusively for academic and research purposes and comply with their respective licenses and terms of use.

References

[1] Hasanah, U., Cahyana, A. (December 27, 2023). *CheXNet and feature pyramid network: a fusion deep learning architecture for multilabel chest X-Ray clinical diagnoses classification.* Retrieved from PubMed: `https://pubmed.ncbi.nlm.nih.gov/38150139/`

[2] Hui, J. (March 26, 2018). *Understanding feature pyramid networks for object detection (FPN).* Retrieved from Medium: `https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c`

[3] Ito Aramendia, A. (March 1, 2024). *DenseNet: a complete guide.* Retrieved from Medium: `https://medium.com/@alejandro.itoaramendia/densenet-a-complete-guide-84fedef21dcc`

[4] Liz, H., Huertas-Tato, J., Sánchez-Montañés, M., Del Ser, J., & Camacho, D. (n.d.). *Deep learning for understanding multilabel imbalanced Chest X-ray datasets.* Retrieved from ar5iv.labs.arxiv: `https://ar5iv.labs.arxiv.org/html/2207.14408`