Pneumonia Detector

Artificial Intelligence

CPCS-331

26th November 2025

Instructor : Dr. Ahmed Hariri

Members :

Saud Alfhaid 2339863

Faris Alzahrani 2339660

Mohammed Alduhaim 2335061

Abdulrahman Aljahdali 2339650

0.Index

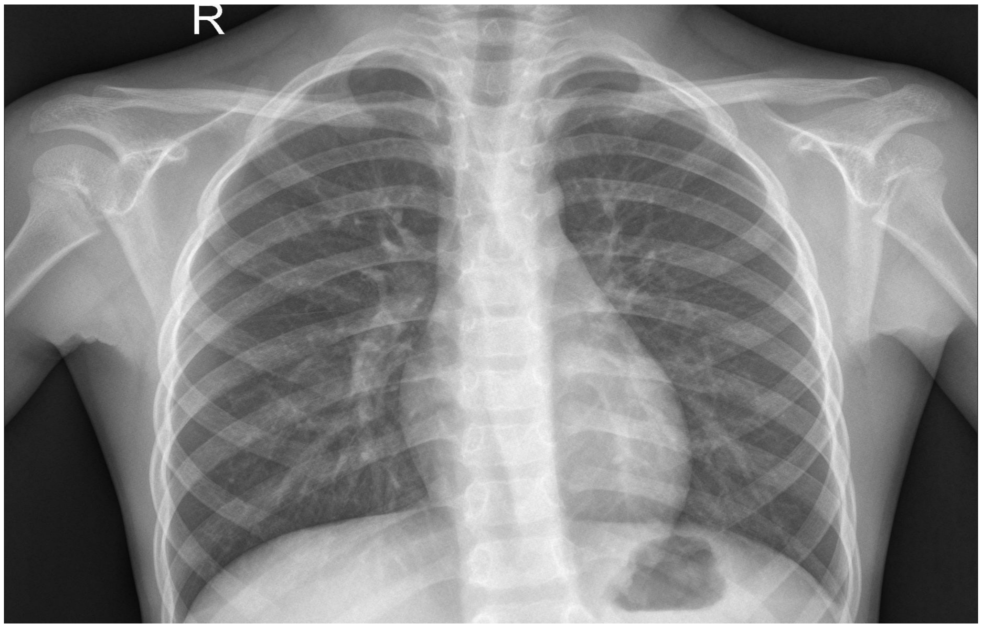
1.Introduction

Pneumonia is a lung infection that can be clearly visible on chest X-ray images, but interpreting these images requires time and expert knowledge. In busy hospitals or in places where radiologists are limited, it can be difficult to quickly and consistently review every X-ray.

The problem we address in this project is to automatically classify chest X-ray images into two categories: Normal or Pneumonia. Given a single X-ray as input, the model decides whether there are signs of pneumonia or not, helping to support doctors in the early detection of the disease.

We chose this problem because it combines a real medical need with practical machine learning techniques.

Pneumonia is a common and serious condition, so improving its detection can have a direct impact on patient care. At the same time, a publicly available dataset of chest X-rays makes it possible to experiment with deep learning models in a realistic setting. Working on this topic allows us to explore how AI can assist healthcare professionals, understand the challenges of imbalanced medical data, and evaluate how well a vision model can learn to recognize subtle patterns in X-ray images.



2.Data

The data for this project comes from the public Kaggle dataset **[1]**, which contains chest radiographs labeled as either *Normal* or *Pneumonia*. The images are already divided into three folders: a training set, a validation set, and a test set. In total, the dataset includes 5,856 – 5,863 X-ray images collected from real patients in a clinical setting. The standard split used in many studies is: 5,216 images for training, 16 images for validation, and 624 images for testing.

Each split has a different distribution of the two classes :

- The training set : clearly imbalanced, with many more Pneumonia images than Normal ones, so the model is exposed to more diseased cases during learning.
- The validation set : very small and roughly balanced and is mainly used to monitor the model while training.
- The test set : larger but still slightly imbalanced, with 234 Normal and 390 Pneumonia images, and is used only for final evaluation.

Because of this imbalance, the model may tend to favor the Pneumonia class, so accuracy alone can be misleading if we do not also look at precision, recall, and the confusion matrix.

To make these differences clearer, the dataset split can be summarized into a table :

Set split	Total Images	Normal	Pneumonia
Training	5,216	1,341	3,875
Validation	16	8	8
Test	624	234	390

This table highlights both the size of each subset and the class imbalance, especially in the training and test sets, which directly affects how the model learns and how we interpret its results. The training set class distribution is also visualized in figure 1, and also include example images from the dataset in figure 2.

2.Data

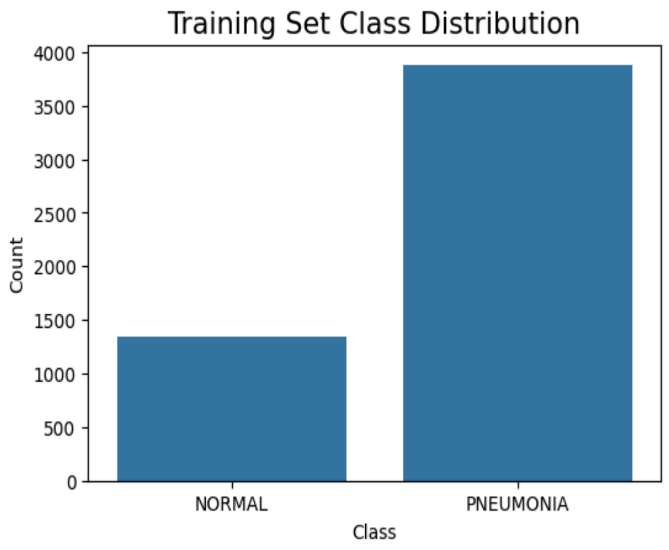


Figure 1

The figure shows that Pneumonia images are much more frequent than Normal images

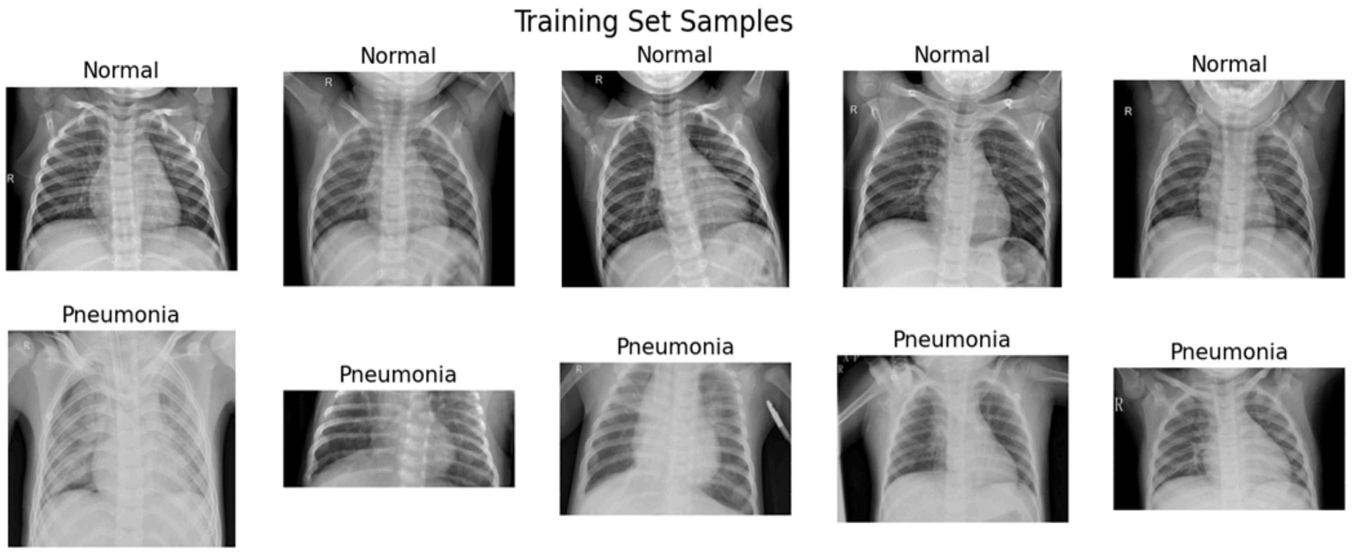


Figure 2

The figure shows Normal chest X-ray and Pneumonia chest X-ray next to each other to illustrate the visual differences in lung appearance, opacity, and texture between healthy and infected lungs

2.Data

All images in the project are preprocessed using *torchvision.transforms* **[3]** so they match the input format expected by an ImageNet-pretrained model and to slightly increase the variety of the training data .

Here is what we have done with the data :

- For the training set, each X-ray is first resized to 256 × 256 pixels and then randomly rotated within a range of −20° to +20°. This random rotation acts as data augmentation and helps the model become less sensitive to small changes in patient position.

- After that, the image is center-cropped to 224×224, converted to grayscale image with 3-channel version using Grayscale method, and then converted into a tensor.

- Finally, the tensor is normalized with the standard ImageNet mean and standard deviation values, where the mean is [0.485, 0.456, 0.406] and the standard deviation is [0.229, 0.224, 0.225 to match what our model were using].

For the validation and test sets, the idea is to keep the preprocessing simple and deterministic. Images are resized directly to 224×224 pixels, converted to grayscale with three channels in the same way as in training, transformed into tensors, and normalized with the same ImageNet statistics.

No random rotation or other augmentation is applied in these splits so that the evaluation reflects the true performance of the model without additional randomness.

The dataset itself consists of clean image files, so there was no need for manual handling of missing values or corrupted entries.

3.Modeling

The type of model we chose for our dataset is EfficientNet-B4, which is a type of convolutional neural network and used as a pretrained model on ImageNet **[4]**

The reason we chose this model is due to the very strong accuracy on image classification while remaining relatively efficient. Since it is pretrained on ImageNet, it already knows useful visual features, which makes it a good starting point for our X-ray pneumonia detection task.

To make it more interesting, we actually fine tuned the model to adapt to our dataset of images, and the reason we used fine tuning instead of starting from scratch is due to many reasons :

- The dataset we are dealing with is actually very little compared to what a real world model needs.
- Training a model from zero would require us a huge amount of dataset, way more with what we are dealing with now.
- A huge amount of time & struggle is saved by instead of starting from scratch, with the promise of having high accuracy & performance.

Let’s talk more about how our models are assured to work with our dataset, but first we need to state that the model, EfficientNet-B4, is separated into two parts :

- Backbone : sees the image / data itself and extracts the features
- Head : Takes those features & works on the decision making

First for the backbone, we made sure that the images were compatible with it. The reason we worked on this is because our model accepts the following :

- RGB images.
- 224 square resolution.
- Images normalized with mean and standard deviation compatible with ImageNet.

As you can see, our dataset is quite the opposite of all of these criteria, so what have we done? What we had to do is the following :

- Converted to grayscale with three channels to match the model.
- Resized them to 244 square resolution.
- Normalized all of them.

3.Modeling

Second for the head, we made sure that the outputs matched our labels. The reason we needed to assure this is because the model we are using is trained to output thousands of classes while we only need two classes, which are Pneumonia and Normal.

So the actions we took to assure that everything is labeled clearly to our needs are :

- Changed the 1000 class classifier with a linear layer
- The linear layer outputs one number that is applied to sigmoid
- If it is close to 1, Pneumonia. Else it is Normal

In short, the detector does the following :

1. Takes the dataset, and manipulate it to the backbones liking
2. Takes the features from the data then makes the decision
3. The decision is decided by a number generated by a linear layer that is placed into a sigmoid
4. If the number is more than 0.5, close to 1, then it would treat it as Pneumonia. Else it would treat it as Normal lungs.

The model is trained using PyTorch **[2]**, we use the AdamW optimizer with a learning rate of 0.001 and we train for 20 epochs.

During the epochs, the accuracy quickly reaches very high values, around 98–100%, and then remains relatively stable. However, for the validation dataset itself, due to the low number of data we got and the number of epochs done, the accuracy was oscillating very fast.

In each epoch, we train the model step by step on small groups of images, let it learn from its mistakes, and check on a separate set if it’s actually learning and not just memorizing :

1. The model runs a forward pass to make predictions.
2. The loss is computed by comparing predictions to the true Normal / Pneumonia labels.
3. The gradient is computed and the model’s parameter is updated.
4. After the epochs, we compute validation loss and accuracy.

Because we have more Pneumonia images than Normal ones, we don’t just trust accuracy, we also use augmentation and detailed metrics to make sure the model is fair and not biased toward Pneumonia only.

3.Modeling

The model is evaluated on the held out test set of 624 images (234 Normal, 390 Pneumonia), which is kept hidden during training or validation. On this split we compute standard metrics for binary classification each with reasoning why we chose them :

- **Accuracy** – Gives a quick overall view of how many X-rays were classified correctly out of all test images.
- **Precision** – Tells us how reliable a positive Pneumonia prediction is and it controls false positives and useless alarms
- **Recall** – Tells us how many real Pneumonia cases the model actually catches and it is very important so we don't miss sick patients.
- **F1-score** – Balances precision and recall in one value, which is more meaningful than accuracy alone on an imbalanced dataset.
- **Classification report (per class)** – Shows precision, recall, and F1 separately for Normal and Pneumonia, so we can see if the model is good on both classes, not just the majority one.
- **Confusion matrix** – Shows the exact counts of correct and incorrect predictions (TP, TN, FP, FN), helping us see whether the model makes more dangerous errors (missed Pneumonia) or safer ones (false alarms).

On this test set, the model reaches a **test loss of 0.2187** and an **accuracy of 94.71%**. Here is a table with all the important percentages of performance for each metric :

Category	Precision	Recall	F1	Support
Overall	0.9519	0.9641	0.9580	624
Normal	0.94	0.92	0.93	234
Pneumonia	0.95	0.96	0.96	390

3.Modeling

The confusion matrix for the test set is shown in **Figure 3**. It confirms the predictions that **false positives** (Normal → Pneumonia) are more common than **false negatives** (Pneumonia → Normal). From a medical perspective, this bias is acceptable, since it is safer to flag a healthy image as suspicious than to miss an actual pneumonia case.

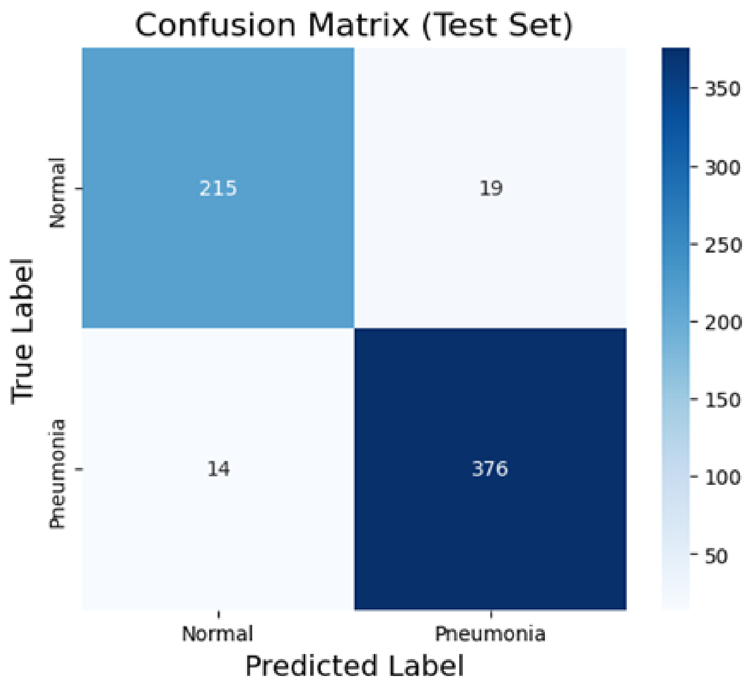


Figure 3

215 Normal images correctly classified as Normal, 19 Normal images predicted as Pneumonia, 376 Pneumonia images correctly classified as Pneumonia, and 14 Pneumonia images predicted as Normal.

4.Conclusion

Pneumonia is a serious and common lung infection, and its diagnosis often depends on careful reading of chest X-ray images.

In crowded or resource-limited hospitals, an automatic tool that can quickly flag suspicious scans can help reduce the workload on radiologists and support earlier treatment decisions.

In this project we used the Kaggle *Chest X-Ray Images (Pneumonia)* dataset **[1]** and trained an EfficientNet-B4 model **[4]** with transfer learning in PyTorch **[2]**.

After preprocessing and training for 20 epochs, the final system achieved a test loss of 0.2187 and an accuracy of 94.71% on a test made of 624 hidden X-rays,.

The model shows especially strong performance on the Pneumonia class, with high recall and relatively few missed positive cases, while the confusion matrix indicates that some Normal images are classified as Pneumonia, which is a safer mistake than missing an actual infection.

Here are some of the references integrated /used for our project :

[1] *P. Mooney, "Chest X-Ray Images (Pneumonia)," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia Kaggle*

[2] *A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32 (NeurIPS 2019), 2019. [Online]. Available: https://arxiv.org/abs/1912.01703 arXiv+1*

[3] *TorchVision maintainers and contributors, "TorchVision: PyTorch's Computer Vision library," GitHub repository, 2016. [Online]. Available: https://github.com/pytorch/vision GitHub*

[4] *M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. 36th Int. Conf. Machine Learning (ICML), 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html*