Universidad Francisco de Vitoria

Computer Science and Engineering

Second Semester / Course 2024/2025

Artificial Intelligence 2
Practical Work III

Done by

Christopher Cobo Piekenbrock

Rodrigo Sáez Escobar

Gonzalo Eusa Silvela

Jaime López de Heredia Delgado

Juan Fernández Cerezo

Leader

Gonzalo Eusa Silvela

Teacher

Moisés Martínez Muñoz

Abstract

This practical work focuses on using deep learning to classify chest X-ray images into three categories: normal lungs, bacterial pneumonia, and viral pneumonia. We used the Chest X-Ray Images (Pneumonia) dataset and applied two different approaches. In the first exercise, we built a convolutional neural network (CNN) from scratch. In the second exercise, we used transfer learning with the ResNet50 model. Both models were trained, evaluated, and compared using the same dataset. The goal was to test different techniques, understand the impact of data preparation and augmentation, and identify which model works better for this classification task. The results show that both models achieved good performance, with the CNN-based model slightly outperforming the transfer learning approach.

Table of Contents

Table of Figures

Index of Tables

1. Introduction

In this project, we learned how deep learning can help with the detection of pneumonia by analyzing chest X-ray images.

Our work is divided into two exercises. In Exercise 1, we built a convolutional neural network (CNN) from scratch using Keras. In Exercise 2, we applied transfer learning by using a pre-trained model (ResNet50) and adapting it to our task. Both models were trained and tested using the same dataset, so we could compare their performance in a fair way.

We paid special attention to data preparation and data augmentation, as well as how different model architectures and training techniques affect the results. One of the key goals was to see which model performs better and generalizes best: the CNN built from scratch or the one based on transfer learning.

All experiments done are influenced by the learnings obtained in class and with the slides provided by our teacher from Unit 6.1 - Unit 6.1 - Deep Learrning applications (Computer Vision) [1]

2. Exercise 1

2.1. Description of the exercise and their goals.

The main goal of this exercise is to develop a deep learning model that can classify chest X-ray images into three categories: viral pneumonia, bacterial pneumonia, and no pneumonia (Normal).

Originally, the dataset was designed for binary classification (pneumonia vs. normal), so before starting, we had to restructure the dataset to adapt it for a multi-class classification problem. This involved separating the pneumonia cases into two groups (Virus and Bacteria), and then generating new training, validation, and test sets with a balanced distribution.

An important part of this project is learning how to prepare and clean the input, that means, the image data. Before training any model, we applied several preprocessing steps that we will explain later.

But mainly, firstly we checked corrupted files, we analyzed all image resolutions and decided to resize all images to (152, 230). As we have mentioned, we reorganized the dataset into three folders (train, val, and test) and made sure all three classes were represented in each one. Finally, we tested whether data augmentation could help the model improve its generalization.

The model we built is a Convolutional Neural Network (CNN) using Keras with all the instructions that we had in the statement.

To fully evaluate our approach, we trained and compared two versions of the model: one with data augmentation, and one without augmentation.

At the end, we selected and evaluated the best performing model based on validation accuracy and tested it on unseen data. This allowed us to understand not only how to build a model, but also how important is a good data preparation is, in many cases more important than building a good model.

2.2. Description of the Dataset

The dataset used in this exercise is the Chest X-Ray Images (Pneumonia) dataset, obtained from Kaggle. It consists of a total of 5,856 grayscale chest X-ray images, each with an approximate resolution of 1152x760 pixels.

The dataset used in this exercise is Chest X-Ray Images (Pneumonia) from Kaggle, which contains 5,856 grayscale chest X-ray images. Each image belongs to one of three categories: bacterial pneumonia, viral pneumonia, or normal.

Although the dataset was originally designed for binary classification, we adapted it to support a multi-class classification task by separating pneumonia cases into bacterial and viral groups.

2.3. Data Preparation Methods

Before training the model, we carefully checked and organized the dataset. This step was very important to avoid errors and to make sure the model learned from clean and correct data.

First, we checked that all the images were valid, and we removed any corrupted files using a script. We also deleted unnecessary system files like .DS_Store, which can cause problems when reading the folders.

Next, we analyzed the image sizes and found that they had many different resolutions. To make everything easier for the model, we decided to resize all images to (152, 230). This size helped us keep the important details while training faster.

The original dataset was made for a binary classification task. So, we split the PNEUMONIA folder into two new classes: VIRUS and BACTERIA, based on the image filenames. Then, we reorganized everything into three folders: train, val, and test, and made sure that each class had images in all of them.

After that, we used code to move 15% of the images to the test set and 5% to the validation set for each class. We did this randomly to make sure the split was fair.

We also calculated class weights because some classes had more images than others. These weights helped the model not to "ignore" the classes with fewer images during training.

And finally, we created two types of data generators. As we have seen the importance in the class theory of the augmentation, we thought of making one with data augmentation (rotations, shifts, zooms...) for training, to increase variety and avoid overfitting and the other one without data augmentation to compare and see the real effect of augmentation.

2.4. Experimentation and Evaluation

Once the dataset was fully prepared, we started experimenting with different models to classify the X-ray images.

We first knew how many images were in each folder (train, validation, and test) and in each class (Virus, Bacteria, and Normal). This gave us a clear idea of how the data was distributed and if everything we have done previously was alright.

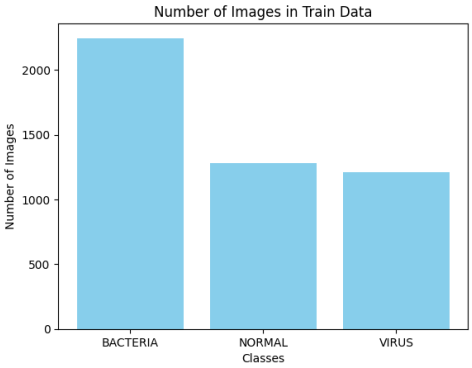

Figure 1-Images for training (4731 Images) - Self-made

4731 images for training (Figure 1)

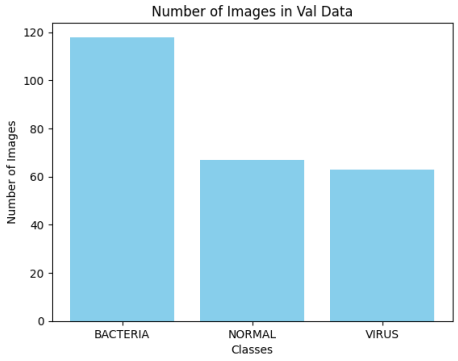

Figure 2-Images for validation (248 Images) - Self-made

248 for validation (Figure 2)

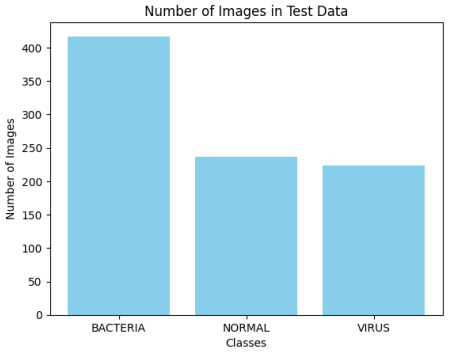

Figure 3-Images for testing (877 Images) - Self-made

877 for testing (Figure 3)

We also calculated class weights to help the model handle the class imbalance better during training.

```
Class 'BACTERIA': 0.7024498886414254
Class 'NORMAL': 1.2329945269741986
Class 'VIRUS': 1.3065451532725767
```

Figure 4-Weights for each class – Self-made

These values show that the NORMAL and VIRUS classes had fewer images than BACTERIA. Because of that, the model paid more attention to them during training. This helped the model do better when trying to detect VIRUS cases.

After that, we built and tested two versions of our model. One with data augmentation, where the training images were randomly rotated, flipped, zoomed, and shifted. And the other one without data augmentation, using only the original images.

For both versions, we tested three different configurations of the CNN model.

```
configurations = [
    {"kernel_sizes": [(5, 5), (3, 3), (2, 2)], "num_layers": 3, "learning_rate": 0.0005},
    {"kernel_sizes": [(7, 7), (5, 5), (3, 3), (2, 2)], "num_layers": 4, "learning_rate": 0.0001},
    {"kernel_sizes": [(7, 7), (5, 5), (3, 3), (3, 3), (2, 2)], "num_layers": 5, "learning_rate": 0.00005},
]
```

Figure 5- CNN Configurations Used for Training - Self-made

As you can see in figure 5, each configuration changed the number of convolutional layers, the kernel sizes, and the learning rate. All models used Relu as the activation function in the convolutional layers and softmax at the end to classify the images.

Each model was trained for 100 epochs using a batch size of 32 and images of size (152, 230). We also used the validation set to monitor how well each model was learning.

After training all models, we selected the best one from each version (with and without augmentation), based on the best validation accuracy. Then, we tested both of them using the test set, which contains images the model had never seen before.

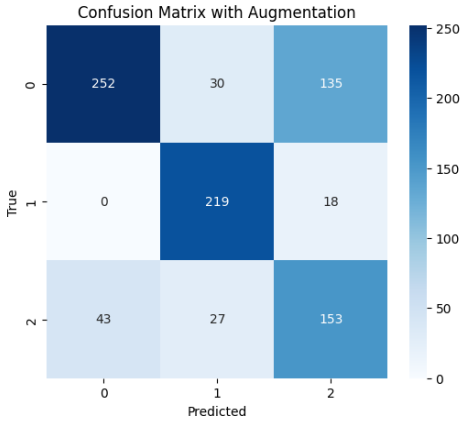

Figure 6-Confusion Matrix with Augmentation – Self-made

	precision	recall	f1-score	support
BACTERIA	0.85	0.60	0.71	417
NORMAL	0.79	0.92	0.85	237
VIRUS	0.50	0.69	0.58	223
accuracy			0.71	877
macro avg	0.72	0.74	0.71	877
weighted avg	0.75	0.71	0.71	877

Figure 7-Metrics to evaluate different classes – Self-made

As you can see in figure 6 and figure 7 when we trained the model using data augmentation, the total test accuracy was 71%. As you can see, the model worked very well for the Normal class (recall = 0.92) and quite well for Bacteria (precision = 0.85). However, it had some trouble with the Virus class. It missed many virus cases (recall = 0.69) and often confused them with bacteria or normal.

This shows that data augmentation helped a bit for Normal, but didn't improve much for Virus. In fact, Bacteria predictions were less accurate when using augmentation.

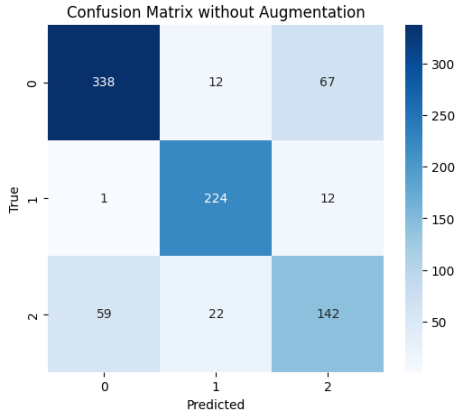

Figure 9- Confusion Matrix without Augmentation - Self-made

	precision	recall	f1-score	support
BACTERIA	0.85	0.81	0.83	417
NORMAL	0.87	0.95	0.91	237
VIRUS	0.64	0.64	0.64	223
accuracy			0.80	877
macro avg	0.79	0.80	0.79	877
weighted avg	0.80	0.80	0.80	877

Figure 8-Metrics to evaluate different classes – Self-made

When training without data augmentation, the test accuracy improved to 0.80. The model performed better across all three classes, with a much higher recall for Bacteria (0.81) and slightly better for Virus (0.64). The confusion matrix shows fewer errors, especially for the Bacteria class.

So, in our case, the model without augmentation worked better. This surprised us a little, because normally augmentation helps. But maybe in this dataset, it could be because the augmented images added too much noise, or because the original dataset was already big enough.

Finally, we evaluated the best model. As we have worked with medical images, we investigated that in medical problems calculating the recall (sensitivity) and specificity for the VIRUS class was very important.

```
Recall (sensibilidad): 0.6367713004484304
Specificity (especificidad): 0.6425339366515838
```

Figure 10- Sensitivity and Specificity for the VIRUS Class- Self-made

As you can see in figure 10, the recall is 0.64. This means that the model correctly found 64% of the real virus cases. It is not perfect, but it's useful for catching most of the dangerous cases. And the specifity also 0.64. This means that the model correctly avoided false alarms (false positives) in 64% of the cases when the patient did not have the virus.

These results show that the model is balanced.

Something important for you to know : in our code, we included a comment explaining that before choosing the final configurations (image size = 152×230, filters = 16×2^i), we actually tested many different setups.

2.5. Top Three Predicted Categories (Example Predictions)

To better understand how our model works, we visualized some examples from the test set. For each image, we show the true label and the top 3 predictions made by the model, sorted by probability.

Below are three examples: one for each class (VIRUS, BACTERIA, NORMAL):

Figure 11-VIRUS prediction Image – Self-made

Figure 12- BACTERIA prediction Image – Self-made

Figure 13- NORMAL prediction Image – Self-made

In the first image, the true label is Virus, and the model predicted it correctly with high confidence (95%). In the second image, the true label is Bacteria, and the model also got it right, but with less confidence (56% vs. 44% for Virus). In the third image, the true label is Normal, but the model confused it and gave the highest score to Bacteria (46%).

These results show that while the model is often correct, it sometimes struggles, especially when the differences between classes are subtle.

The rest of the predictions are available in the notebook and can be reviewed to better understand the strengths and weaknesses of the model.

2.6. Questions

2.6.1. Has your model achieved optimal training?

Our model reached a good level of performance, but we don't think it was fully optimal. During the process, we tested many different things: we added more data, changed the input size, increased the number of filters, changed the number of steps per epoch, and tried early stopping. We also tried different ways to prepare the dataset before deciding on the final version.

In the end, we focused on comparing two versions: one with data augmentation and one without it. The best results came from the model trained without augmentation, which got around 80% test accuracy. It performed well on Normal and Bacteria, but the scores for Virus were lower.

This means the training worked quite well, but there is still room to improve, especially for detecting virus cases.

2.6.2. What does the confusion matrix reveal about your model's performance?

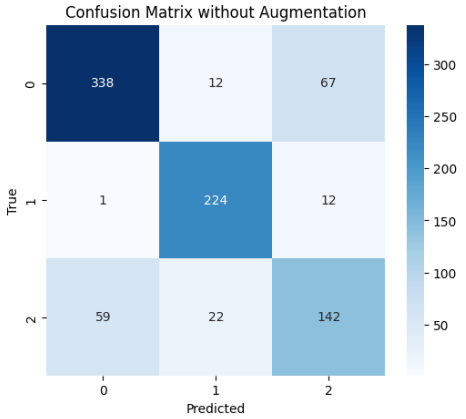

Figure 14- Confusion Matrix without Augmentation – Self-made

	precision	recall	f1-score	support
BACTERIA	0.85	0.81	0.83	417
NORMAL	0.87	0.95	0.91	237
VIRUS	0.64	0.64	0.64	223
accuracy			0.80	877
macro avg	0.79	0.80	0.79	877
weighted avg	0.80	0.80	0.80	877

Figure 15- Metrics to evaluate different classes – Self-made

The confusion matrix and the classification report show that the model performs very well overall, especially on the Normal and Bacteria classes. It achieves high recall for Normal (0.95) and strong results for Bacteria (recall = 0.81, precision = 0.85), meaning it is very good at identifying these two types of cases.

The Virus class is more challenging, with both precision and recall at 0.64. This means the model sometimes confuses viral pneumonia with the other two categories. However, considering the difficulty of distinguishing between these classes in medical images, the results are still quite solid and consistent.

In general, the model shows strong performance across all categories, and only minor improvements are needed for better detection of the Virus class.

2.6.3. How did data preprocessing impact the model's performance?

We already explained in previous sections how important preprocessing was in our project. From checking corrupted images to reorganizing and balancing the dataset, every step helped us get better results.

Thanks to all these process, we managed to build a model that reached 80% accuracy. So in our case, preprocessing was not just useful, it was key.

3. Exercise 2

3.1. Description of the exercise and their goals.

The goal of this exercise was to apply transfer learning to classify chest X-ray images into three categories: Normal (healthy lungs), Bacterial pneumonia, and Viral pneumonia. We used a pretrained model

(ResNet50) and tested how different learning rates and the use of data augmentation affected performance. We also analyzed the model's predictions using accuracy, precision, recall, F1-score, and confusion matrices. This exercise helped us understand the advantages of using pretrained networks.

3.2. Description of the Dataset

This dataset is identical to the one use in exercise 1 and to prepare the dataset for our model, we use the same process as in it. The pneumonia cases were further divided into two categories: Bacterial and Viral based on the filename. The images were already grouped into three folders: train, test, and validation. Each folder contained subfolders for the classes. We followed the same process from exercise 1.

3.3. Data Preparation Methods

The dataset preparation followed the same structure we used in the first exercise, with some adjustments for transfer learning.

All images were resized to 224x224 pixels, since ResNet50 requires a square input of this size.

Although the original images were grayscale, we used RGB (3 channels) because ResNet50 was trained on RGB images (ImageNet). This allowed us to use the pretrained weights without modifying the network structure.

We applied data augmentation during training (random rotations, flips, zoom, and shifts) to help the model generalize better and avoid overfitting.

We also rescaled the pixel values to the range [0, 1].

Finally, we computed and applied class weights to give more importance to underrepresented classes during training, just like we did in Exercise 1.

3.4. Experimentation

We used the ResNet50 model as our base model because of its strong performance in many computer vision tasks. It is a deep architecture with skip connections that help train effectively without vanishing gradients.

The top layer of ResNet50 was removed (include_top=False) and replaced with a GlobalAveragePooling layer, followed by a dense layer with ReLU activation, a Dropout layer (to reduce overfitting), and finally a softmax output layer for multi-class classification as it is mentioned in the statement.

We tested different learning rates: 0.0001, 0.0005, and 0.001, with and without data augmentation. For each configuration, the model was trained for 10 epochs using a small number of steps per epoch to speed up experimentation. The best models were saved using a ModelCheckpoint callback based on validation loss, which avoids overfitting and ensures we keep the best version of the model.

After training, each model was evaluated on the test set. We also generated a confusion matrix and classification report to better understand the performance per class.

3.5. Top Three Predicted Categories (Example Predictions)

To better understand how our model behaves on real data, we selected three chest X-ray images from the test set, one for each class: Bacterial, Normal, and Viral. For each image, we display the top three predictions made by the model along with their confidence scores.

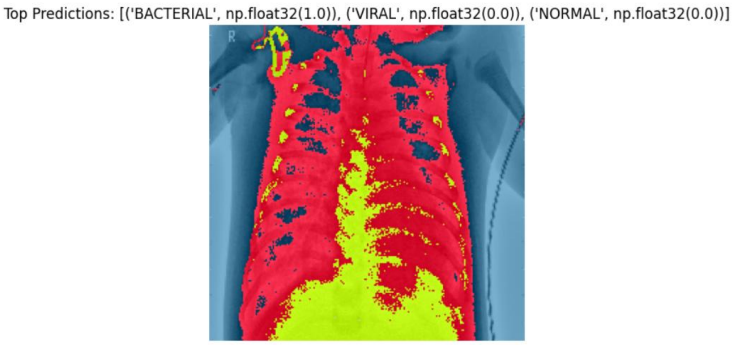

Figure 16- BACTERIAL Prediction – Self-made

In the first image, the true label **is** Bacterial Pneumonia, and the model predicted it correctly with full confidence (100%). The other two predicted classes (Viral and Normal) received a probability of 0%. This indicates that the model was very confident in its decision and likely found strong patterns specific to bacterial infection.

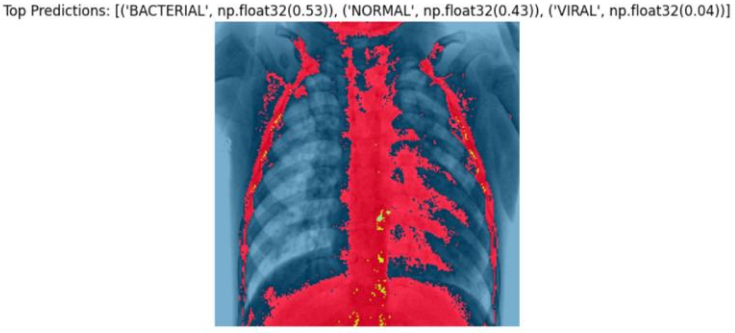

Figure 17- BACTERIAL vs NORMAL – Self-made

The second image also belongs to the Bacterial class. This time, the model still predicted "Bacterial" as the top class, but with lower confidence (53%). It also gave Normal a similar score (43%), and Viral was less likely (4%). This shows a case where the model was uncertain between Bacterial and Normal, suggesting overlapping visual patterns.

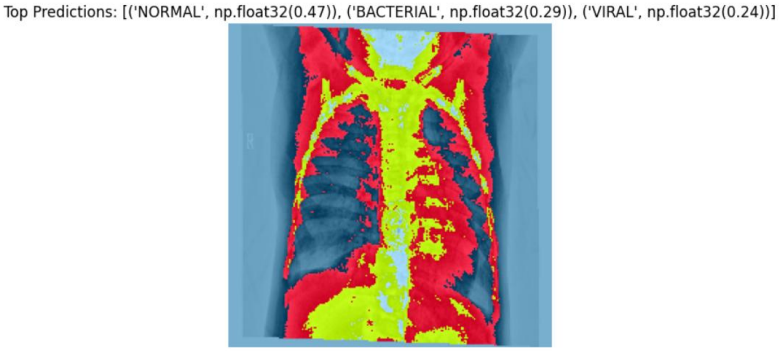

Figure 18- NORMAL Prediction – Self-made

In the third image, the actual label is Normal. The model's top prediction was indeed "Normal" (47%), followed by "Bacterial" (29%) and "Viral" (24%). While the model got the correct class, the scores are close, meaning it was not highly confident and found it visually similar to infected lungs.

These visualizations highlight how the model performs under different levels of certainty. In some cases, such as Figure 11, the model is very confident and correct. In others, like Figures 12 and 13, predictions are more uncertain, which reflects the challenge of distinguishing subtle patterns in chest X-rays.

3.6. Questions

3.6.1. Has your model achieved optimal training?

We believe the model performed well, especially with a learning rate of 0.0005 and no data augmentation, where it reached 79.36% test accuracy and an F1-score of 0.7878. The training and validation metrics showed stable behavior without signs of overfitting, so we consider this training close to optimal.

Learning Rate	Augmented	Accuracy	F1 Score
0.0001	True	0.6796	0.6810
0.0001	False	0.7320	0.7146
0.0005	True	0.7526	0.7508
0.0005	False	0.7936	0.7878
0.001	True	0.7788	0.7757
0.001	False	0.7765	0.7648

Table 1-Different configurations with metrics – Self-made

3.6.2. What does the confusion matrix reveal about your model's performance?

The confusion matrix shows that the model performs well on the Normal and Bacteria classes but still has some confusion between Virus and Bacteria.

For example, in the best model, 121 Virus images were correctly classified, but 89 were misclassified as Bacteria. This highlights a weakness that's not visible through overall accuracy, showing the importance of analyzing per-class performance.

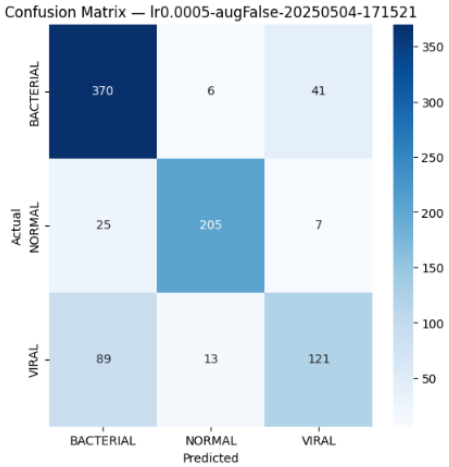

Figure 19-Confusion Matrix without Augmentation (Using pre-trained model) – Self-made

3.6.3. How did data preprocessing impact the model's performance?

Data preprocessing was crucial. Firstly, resizing and using RGB made the dataset compatible with ResNet50. Then, normalizing the pixel values helped the model converge faster. Data augmentation improved generalization in some setups. Computing class weights helped the model better handle class imbalance, especially for the VIRUS class.

These choices had a clear impact on the model's final accuracy and robustness.

4. Comparing results

After finishing both exercises, we compared the results of our own CNN model with the model using transfer learning (ResNet50). The goal in both cases was to classify chest X-ray images into three classes: Normal, Bacterial pneumonia, and Viral pneumonia.

<u>For the CNN model (Exercise 1):</u>

- Accuracy: 80%
- F1-score: 0.80 (weighted)
- Good results for Normal (recall = 0.95) and Bacteria (recall = 0.81).
- Virus class was harder: precision and recall = 0.64.

This model worked very well and we built it from scratch.

And for the Transfer Learning model (Exercise 2 - ResNet50):

- Best accuracy: 79.36%
- Best F1-score: 0.78 (weighted)
- Also worked well for Normal and Bacteria.
- Virus was again the most difficult class.
- Used a pre-trained model with different learning rates.

Even though both models had similar results, our own CNN model got slightly better performance. Also, we built it ourselves and trained it from zero, so it shows that with good preparation and design, we can get strong results without using pre-trained models.

So, we think the CNN model is the best one in this case. It had better generalization and a bit higher accuracy than the transfer learning model.

5. General conclusions

Throughout this project, we realized how important data preparation is in deep learning tasks. Even though model design is relevant, the way we clean, balance, and structure the dataset had the biggest impact on our results. We saw that small decisions like image size, normalization, or class splitting can strongly influence performance.

We also understood the value of evaluating with and without data augmentation. By comparing both strategies in Exercise 1, we saw that augmentation does not always improve results, and it's essential to test different configurations instead of assuming what works best.

In Exercise 2, using a pre-trained model helped us reduce training time and still get strong results, but we confirmed that a well-built model from scratch can perform just as well or even better if the data is well prepared.

6. References

[1] M. M. Muñoz, Unit 6 - Deep Learrning applications (Computer Vision).