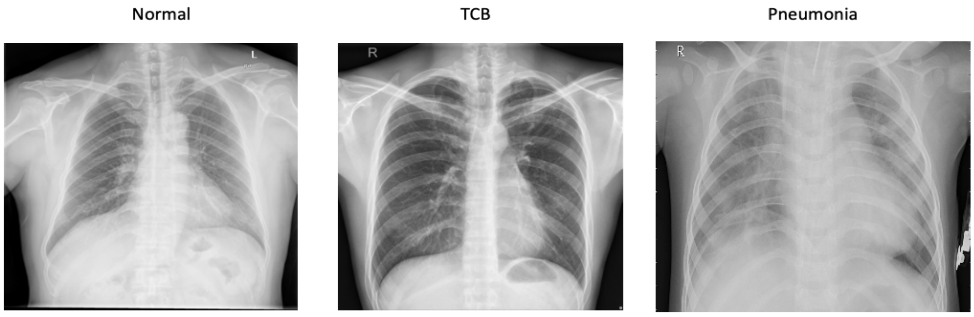
Alessia Lindemann	Michelangelo Olmo Nogara Notarianni	Sara Paratico
M.Sc. Biomedical Engineering	M.Sc. Biomedical Engineering	M.Sc. Biomedical Engineering
ID: 10862854	ID: 10625064	ID: 10626459

AI in Biomedicine: Potential of DL for TB and Pneumonia Detection in Chest Radiography Images

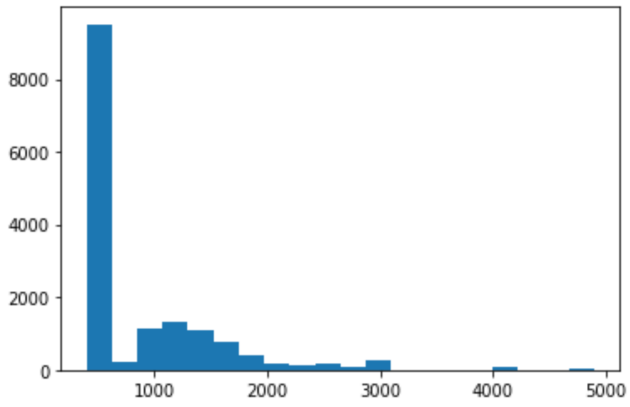
Abstract

Tuberculosis (TB) and pneumonia are two of the most common lung infections, and accurate classification of these diseases from thoracic X-ray images is crucial for early diagnosis and treatment. TB and pneumonia can have similar symptoms, but they are caused by different pathogens and require different treatments. In X-ray images, TB is characterized by the presence of small nodules and cavities, while pneumonia typically shows consolidation of lung tissue. In this study, we aim to develop a deep learning model for tuberculosis and pneumonia classification from thoracic X-ray images with a F1 score greater than 75%. We will use ResNet-like and DenseNet-like models, and we will compare the results obtained through these two approaches. We will also perform explainable AI analysis to understand the model's decision-making process and justify choices in a sensitive way. The goal is to provide a clear interpretation of the results and contribute to the development of more accurate and efficient diagnostic tools for TB and pneumonia.

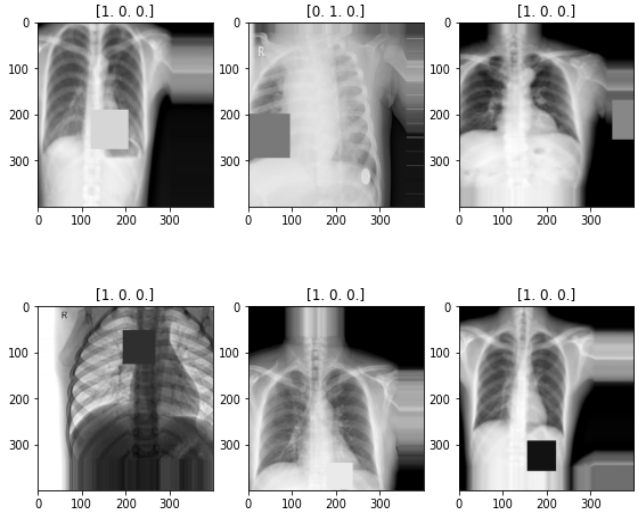


Materials and Methods:

The dataset utilized in this study consisted of 15470 thoracic X-ray images in .png or .jpeg format, which were black and white and had pixels ranging from 0 to 255. The images are divided into three classes: normal (N), pneumonia (P) and tuberculosis (T) with 9354, 4250 and 1866 cases respectively. To ensure that images from the same patient were not present in both the training and testing sets, a stratified train-test-validation split was performed, resulting in a ratio of approximately 0.8, 0.1, and 0.1 for the training, validation, and testing sets respectively. Additionally, k-fold cross-validation was applied to the splitting process with k = 5 for training. Some of the images had large dimensions (more than 4000x4000 pixels), and all of them were squared. The majority of the data had dimensions of 400x400 pixels. Therefore, all images were reshaped to this size.



1: histogram of images resolution (all of them are NxN)



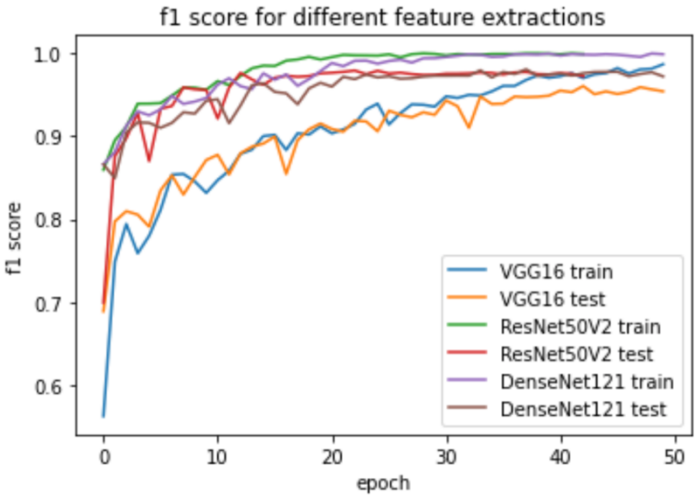
2 Some of the augmented images with their corresponding categorical label [1. 0. 0.] "Normal", [0. 1. 0.] "Pneumonia", [0. 0. 1.] "TB"

To mitigate computational issues, image generators were utilized during the importation process. Since we noticed the presence of poor-quality images, a variety of data augmentation techniques were applied, including random scaling, zoom, shift, shear, and rotation. Additionally, random squared masks were applied to the training set with a given probability. We trained our models to be invariant to these transformations because some of the images in the training set were already noisy, shifted, or partially covered by masks. Also, the augmentation process is meant to reduce the risk of overfitting – the training set size is significantly increased.

At first, we started with a simple, yet effective CNN architecture trained from scratch. It consists of several convolutional layers that extract features from the input greyscale image, followed by fully connected layers learning a decision boundary for the classification task. The number of the filters in the convolutional layers increases as the architecture progresses while max pooling layers reduce the spatial dimension of the feature maps.

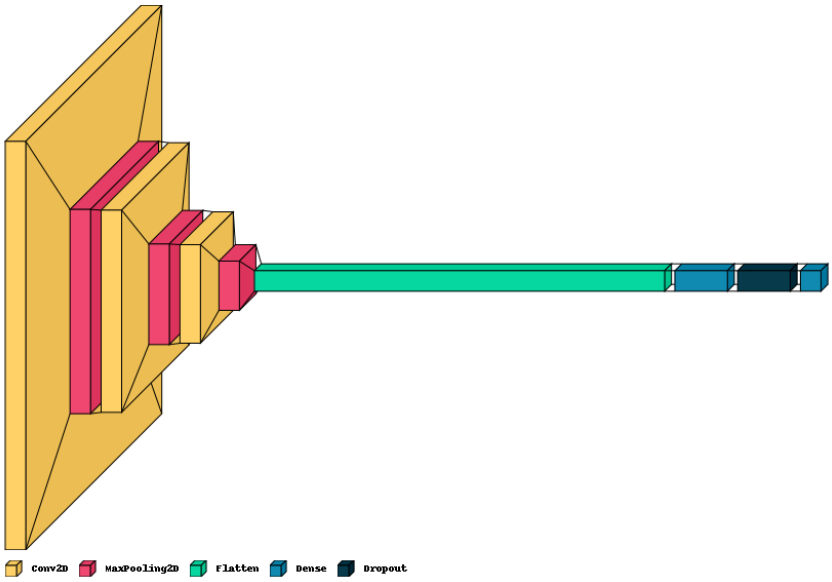
But to handle this limited number of samples, we found better results using transfer learning, i.e., exploiting feature extraction CNNs which were successfully pre-trained on more complex classification tasks (e.g., on ImageNet for 1000 labels).

Given that the pre-trained weights of most of the models we used were trained on RGB images, we often converted the grayscale images to RGB by copying the 2D greyscale matrix in each color channel. Furthermore, a secondary different preprocessing step was applied to the input of the different models. Specifically, input pixel values for the DenseNet model were scaled between 0 and 1, while for the ResNet model, input values were scaled between -1 and 1.
We exploited three popular deep learning architectures as our base models: ResNet50v2, VGG16, and DenseNet121. To ensure robustness of the results, all models were trained multiple times. The pre-trained weights from the ImageNet dataset were utilized for the initial layers of each architecture, with the exception of the output layer, which was randomly initialized using Glorot uniform method. At first, a global average pooling layer was added to each architecture, followed by a fully connected layer with 3 neurons and a softmax activation function to perform the classification task. The Adam optimizer with a learning rate of 1e-4 was utilized for training, along with a batch size of 16 and 20 epochs. Results showed that both ResNet and DenseNet architectures performed best among the base models. We further explored these two architectures, fine-tuning them, adding additional layers, and using Keras Tuner to optimize the hyperparameters such as the number of classifying layers and neurons, the learning rate, and the effectiveness of dropout regularization.



CNN from scratch

Our “vanilla” CNN architecture:

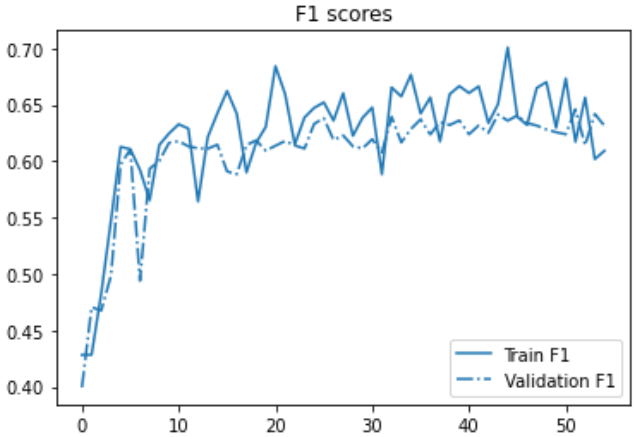


- Starts with a 2D convolutional layer with 32 filters, a kernel size 3x3 and a ‘ReLU’ activation function. This layer will extract features from the input image,
- Adds a max pooling layer with a pool size of 2x2 to reduce the spatial dimensions of the feature maps,
- Repeats these two more times, increasing the number of filters in the convolutional layer by a factor 2 each time (i.e., 64 filters in the second repeat, and 128 filters in the third repeat),
- Adds a flatten layer to convert the feature maps into a 1D feature vector,
- Adds a dense layer with 512 neurons and a ‘ReLU’ activation function. This is a FC layer that will learn a non-linear decision boundary for the classification task.
- Adds a dropout layer with rate of 0.5 to prevent overfitting,
- Adds a final dense layer with 3 neurons and a ‘Softmax’ activation function.

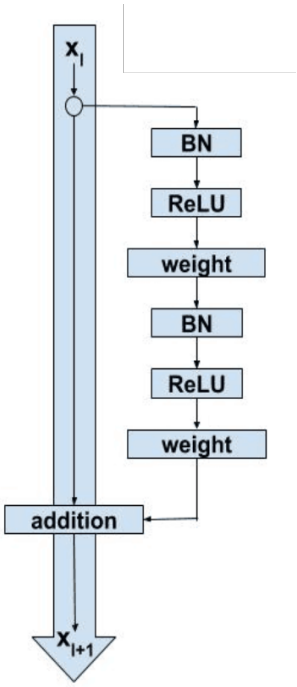
This simple model is similar as networks first used in Deep Learning such as LeNet or AlexNet, which set the foundation for future architectures. In particular, this is a simplified version of AlexNet (introduced by Alex Krizhevsky et al. in 2012) composed by 4 convolutional layers, 1 FC layer and the output layer designed for this case (3 neurons for 3 classes).

It was trained on augmented images, with a weighted loss function, using Adam optimizer with different learning rates. Despite that, it didn’t perform as well as expected (the global f1 score was always below 0.7) and struggled to accurately classify TB. We show here some of the results, which won’t be further discussed. In light of these, we chose to explore the use of pre-trained models with transfer learning as an alternative approach. Their outcomes, shown in the results section, will highlight the importance of leveraging the knowledge gained from other datasets and problems.

Class	precision	recall	f1-score
Normal	0.65	0.98	0.79
Pneumonia	0.87	0.23	0.36
Tuberculosis	0.60	0.04	0.07

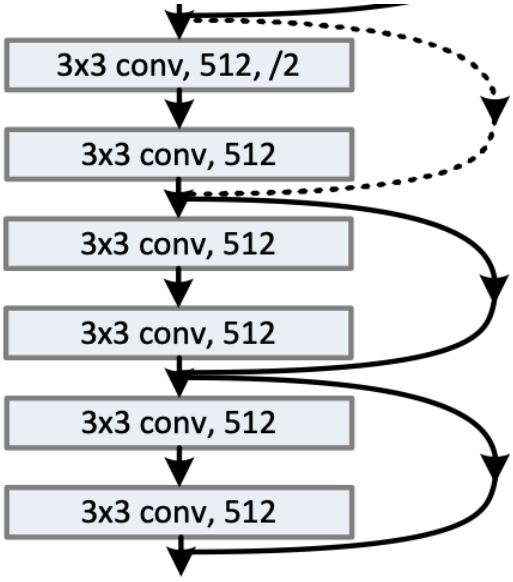


3: Training example: even with class-wise weighted loss, and regularization techniques, results for T are incredibly low. The network is useless for this complex task



Resnet

This table shows the four best results after tuning some hyperparameters on the head of ResNet50V2 with a Bayesian optimization algorithm: the number of layers (0,1 or 2) and the number of neurons in each (either 64, 128, 512, or 1024). We have to say here that to ensure that the results obtained with long hyperparameter tuning are reliable and generalizable to new data, it is important to use techniques such as k-fold cross-validation, which implies dividing the data into k subsets, training the model on k-1 subsets, and evaluating it on the remaining subset. This process should be repeated k times for each model, with each subset serving as the validation set once. The final performance measure is obtained by averaging the performance measures obtained in each fold. This technique allows for a more robust evaluation of the model's performance and can help to identify any overfitting or unrepresentative validation sets.



4 Last ResNet convolutional block; from "Deep Residual Learning for Image Recognition", K.He et al.

no_layers	no_neurons	f1_validation
1	1024	0.9728
1	1024	0.9721
1	1024	0.9611
1	512	0.9600

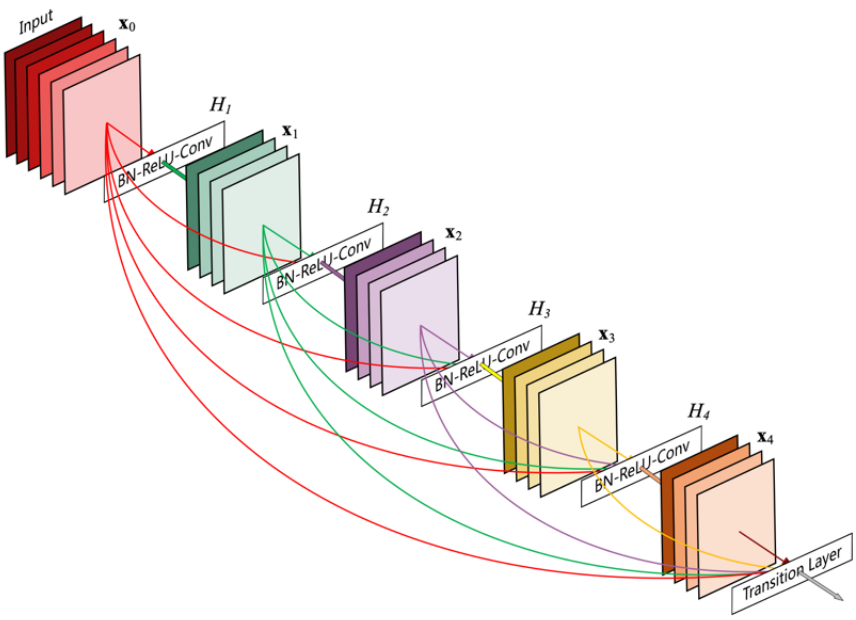
The overall model we used consists of:

- Input layer shaped (400,400,3)
- Preprocessing layer scaling pixel intensities to (-1,1)
- Resnet50V2 as an encoder network / feature extractor: a sequence of 5 convolutional blocks containing sub-blocks of batch-normalization, ReLu activation function and convolutional layers. ResNet stands for Residual Network; it uses skip connections between input and output of each sub-block alleviating the vanishing gradient problem.
- A Global Average Pooling layer, resizing the feature maps (13,13,2048) to 2048 output values.
- (∗) A regularization Dropout layer with rate = 0.2
- A dense layer with 1024 neurons and ‘ReLU’ activation function, initialized with HeNormal technique.
- Three output neurons with softmax activation function, initialized with GlorotUniform technique

In total the network has 25,620,611 trainable parameters. We chose to train it with class-weighted categorical cross-entropy as a cost function; the gradient descent algorithm was optimized with Adam method. The learning rate, initially set to 3e-4, then tuned with Keras Tuner, is optimized with Adam technique, and decreases by a factor of 0.4 whenever the f1-score obtained on the validation set stops improving (4 epochs patience). The training stops after 20 epochs without any f1-score improvement.

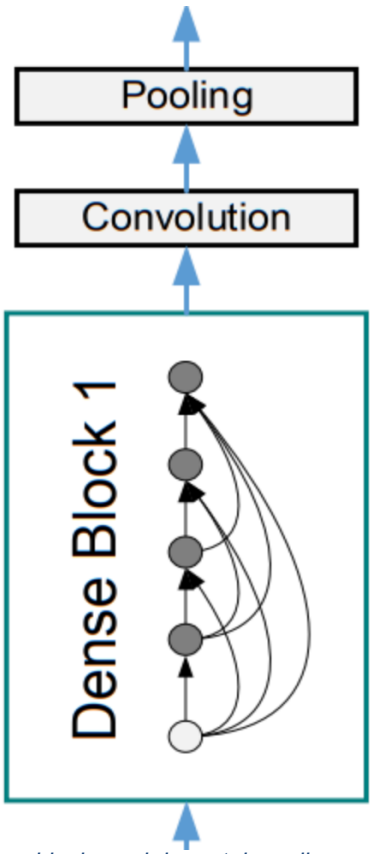
Densenet

We also tried to tune the hyperparameters of the network using Densenet121 as an encoder. This CNN architecture, developed by Gao Huang et al. in 2016, is made of units called “dense blocks”. Each connects all the layers within it directly with each other, allowing feature maps from all previous layers to be used in the current layer, increasing the flow of information through the network. With this base model we found our best result adding a Global Average Pooling layer to reduce the spatial dimension of the feature maps and then a Softmax output. In this configuration with no additional dense layers, the model has fewer parameters and may be less prone to overfitting; still, it was able to capture the same level of complex relationships in the data as Resnet50V2 did.



The overall model we used consists of:

- Input layer shaped (400,400,3)
- Preprocessing layer scaling pixel intensities to (0,1)
- Densenet121 as an encoder network / feature extractor: a sequence of 121 layers divided in sub blocks of several convolutional layers. Between these, transition layers reduce the spatial dimensions applying a 1x1 convolution followed by a 2x2 average pooling operation
- A Global Average Pooling layer, resizing the feature maps (12,12,1024) to 1024 output values.
- Three output neurons with softmax activation function, initialized with GlorotUniform technique



5 In a dense block, each layer takes all preceding feature-maps as input. Images from "Densely Connected Convolutional Networks", G.Huang et al.

In total the network has 6,956,931 trainable parameters (almost ¼ of the parameters in ResNet!). We chose to train it with class-weighted categorical cross-entropy as a cost function; the gradient descent algorithm was optimized with Adam method.

The learning rate, initially set to 5e-4, decreases by a factor of 0.4 whenever the f1-score obtained on the validation set stops improving (4 epochs patience). The training stops after 20 epochs without any f1-score improvement.

Evaluation metrics

Metrics are a crucial part of any machine learning project, and it is essential to choose the appropriate metrics to evaluate the performance of the models. In this study, we used standard metrics such as F1 score, accuracy, precision, and recall evaluating the performance of our models.

- F1 Score: it is a measure of a model's accuracy, calculated as the harmonic mean of precision and recall, where a higher score indicates better performance. In our study, we used the F1 score as the primary metric for evaluating the performance of our models, as it provides a balance between precision and recall.

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

- Accuracy: it is a measure of how well a model correctly predicts the class labels. It is calculated as the ratio of the number of correct predictions to the total number of predictions. However, accuracy can be misleading when the dataset is imbalanced, which is the case in this study.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: it is a measure of how well a model correctly predicts the positive class. It is calculated as the ratio of the number of true positive predictions to the total number of positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: it is a measure of how well a model detects the positive class. It is calculated as the ratio of the number of true positive predictions to the total number of actual positive instances.

$$Recall = \frac{TP}{TP + FN}$$

The primary metric for evaluating the models' performance during training was the global F1 score, which was monitored using callbacks (ReduceLROnPlateau, EarlyStopping). However, to gain a more detailed understanding of the model's performance, and also to accomplish our 75% task, we also evaluated the scores obtained per-class, which allowed us to identify any potential issues with class imbalance.

Explainable AI (XAI)

In addition to traditional evaluation metrics such as accuracy, F1 score, recall, and precision, we also employed explainable AI (XAI) techniques to gain a deeper understanding of our models' decision-making process. These techniques are model-agnostic - thus can be applied to any type of model - and "post-hoc" - applied after the model is trained. However, they have different goals and different ways of interpreting the predictions.

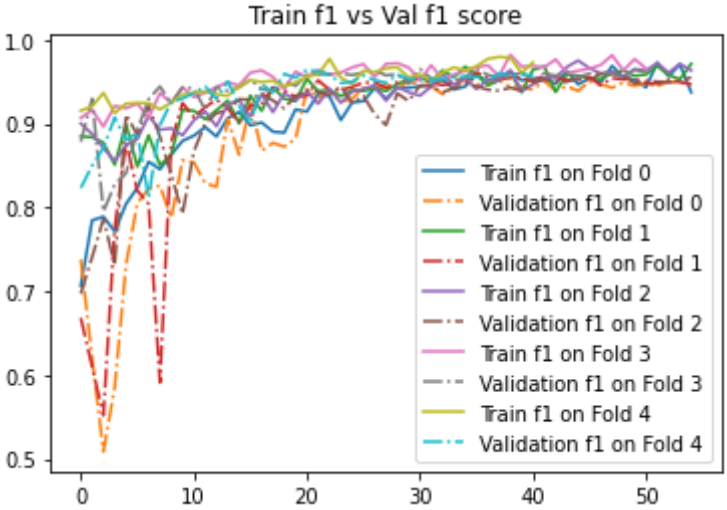
Fairness: The Top-2 Differences method is a way to extend the interpretability of a neural network's predictions by quantifying the uncertainty degree. This is done by comparing the differences between the two highest scores in the Softmax outputs (a categorical distribution [a, b, c] with $a+b+c = 1$, for each image). E.g., the model is "sure" that the chest in the image is "normal" with an output vector of [0.98, 0.015, 0.005], (T2D = 0.965 and the classification is considered reliable). This can help identify which cases are more likely to be misclassified (e.g., predicted "T" with output [0.2, 0.36, 0.44], T2D = 0.08). Moreover, we can measure a global performance of the predictions: one way is to plot a histogram of the differences between the two highest scores for both correctly and misclassified images. The histogram would ideally-hopefully show that for correctly classified images, the majority of the differences are close to 1, indicating high confidence in the predictions. On the other hand, for misclassified images, the differences should be more distributed in the range of 0-1, indicating lower confidence in the predictions.

Grad-CAM: Gradient-weighted Class Activation Mapping is a technique used to visualize which regions of an image were most important for a convolutional neural network's predictions. It is used to understand how a model is making its decisions by highlighting the regions of the image that were most important for a particular prediction. Mathematically, Grad-CAM works by first computing the gradient of the score of a particular class with respect to the feature maps of the last convolutional layer. These gradients are then pooled by taking the average of the gradients over all spatial locations of the feature maps. The resulting pooled gradients are then upsampled to the same spatial resolution as the input image and multiplied element-wise with the feature maps of the last convolutional layer. The resulting heatmap highlights the regions of the image that are most important for the model's predictions.
SHAP: SHapley Additive exPlanations assign each pixel an importance value for a particular prediction, by considering not just the individual contribution of each pixel, but also how they interact with each other. It's like trying to understand how each rational player in a game contributes to the final outcome considering the outcomes of all possible coalitions: Shapley values provide a way to fairly distribute the outcome among a group of players/pixels.
LIME: Local Interpretable Model-Agnostic Explanations is a technique that allows us to understand the predictions of any model by approximating its behavior locally around a specific prediction. This is done by randomly perturbing the input data and observing how the model's output changes. The perturbations are chosen to be close to the original input, allowing to approximate the model's behavior in the neighborhood of the input point. The local interpretable model is trained on the perturbed data and its coefficients give an indication of the feature importance for the specific prediction. This way, LIME provides an explanation of the model's prediction by identifying the features/pixels that have the greatest effect on the output.

Results

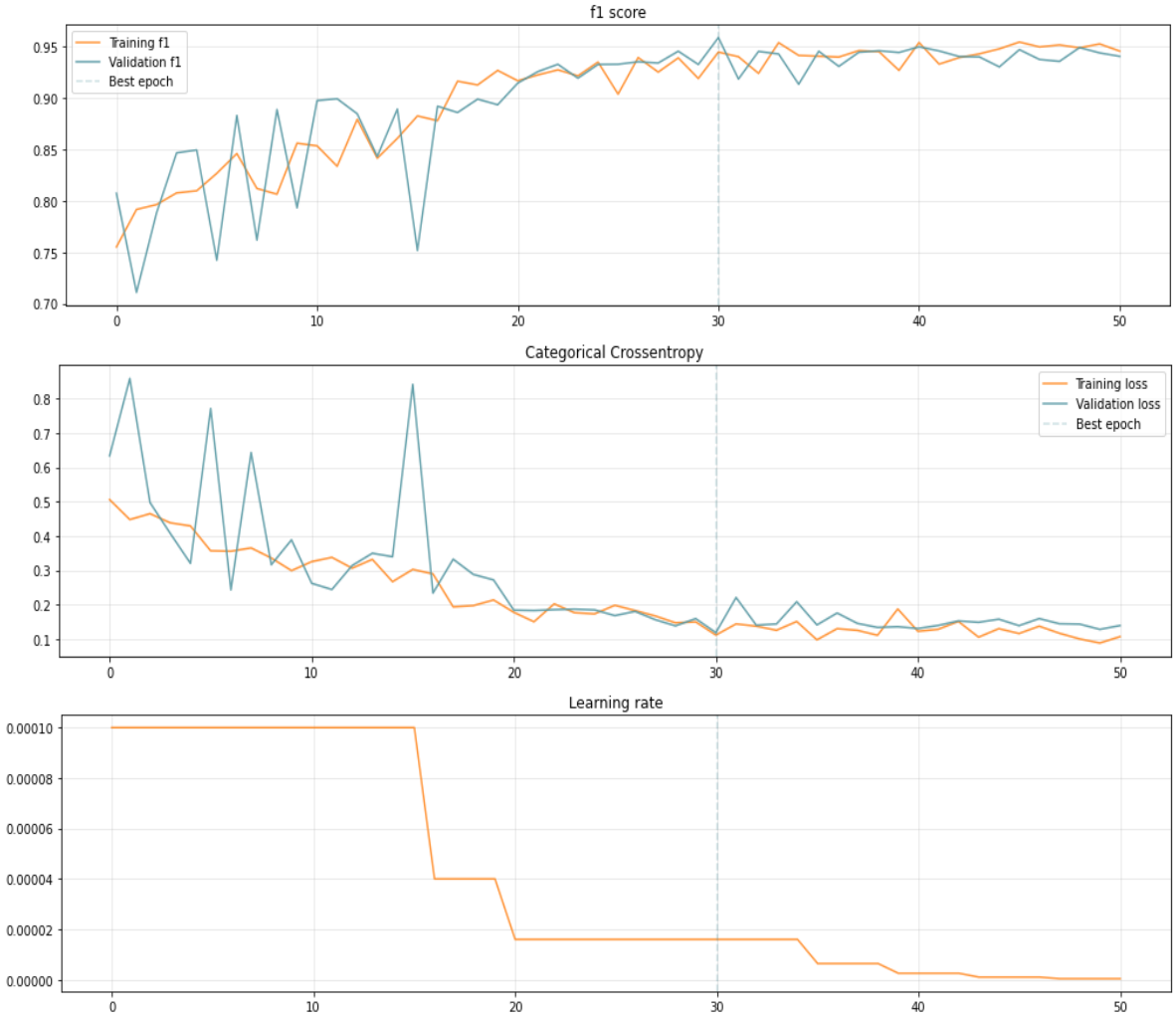
1) ResNet with no dropout

Here we report some of the results obtained with k-fold cross validation techniques to verify the architecture and hyperparameters suggested by tuning with Bayesian optimization algorithm. The initial learning rate was set to 1e-4, is adapted for each layer following Adam optimization algorithm.

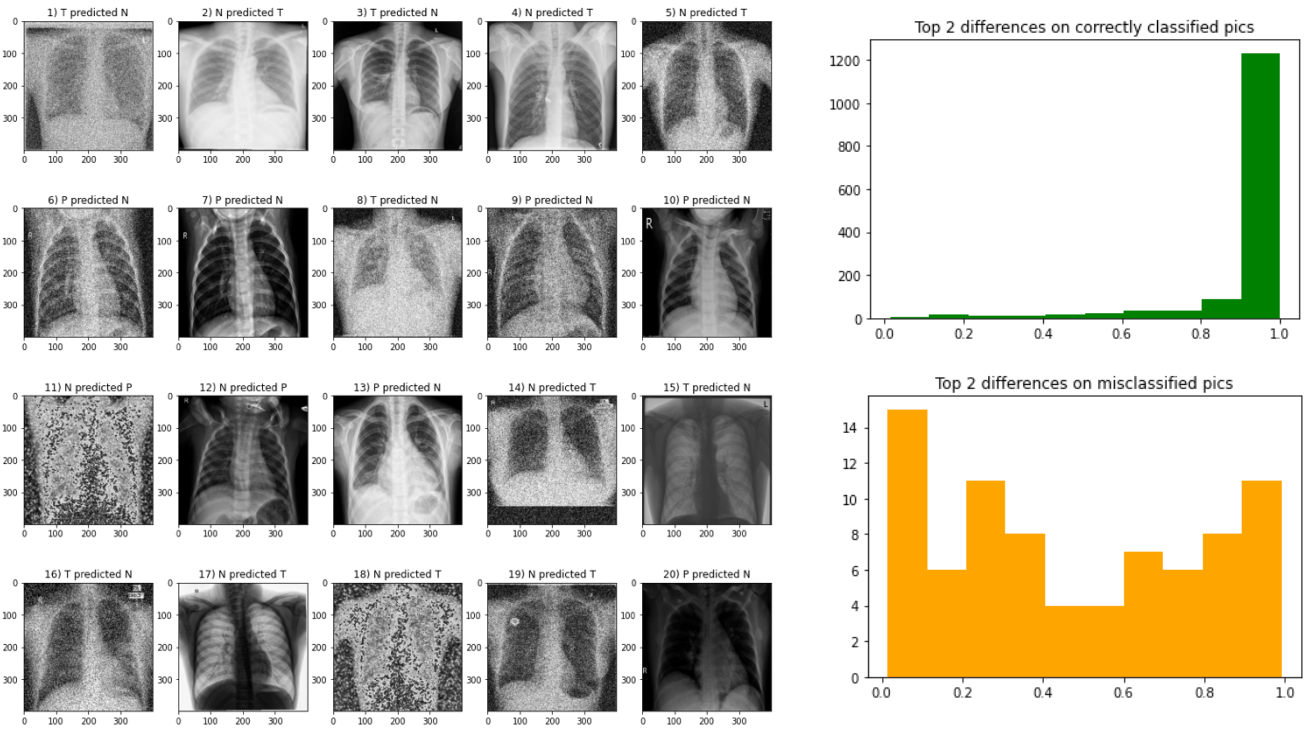


kCV results	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Mean
F1 score	0.9531	0.9611	0.9599	0.9653	0.9645	0.9607
Accuracy	0.9506	0.9585	0.9607	0.9653	0.9642	0.9599
Precision	0.9516	0.9588	0.9609	0.9659	0.9645	0.9603
Recall	0.9492	0.9585	0.9607	0.9650	0.9642	0.9595

Table 1: k-fold cross-validation results on validation subsets with k = 5

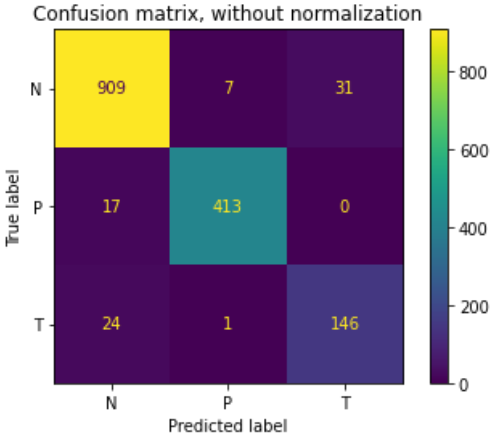


Example: Best f1 score obtained on the validation set (considering all the classes): 0.95892, reached at epoch 30. 80 prediction errors out of 1548 test images.

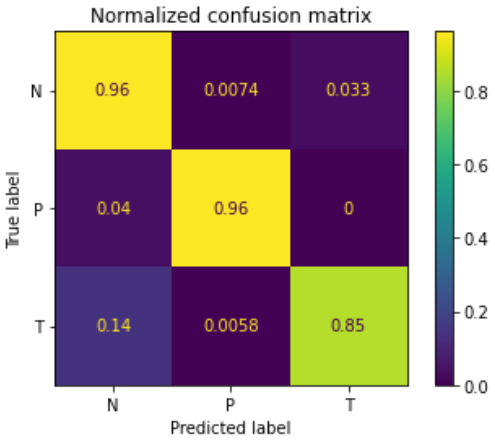


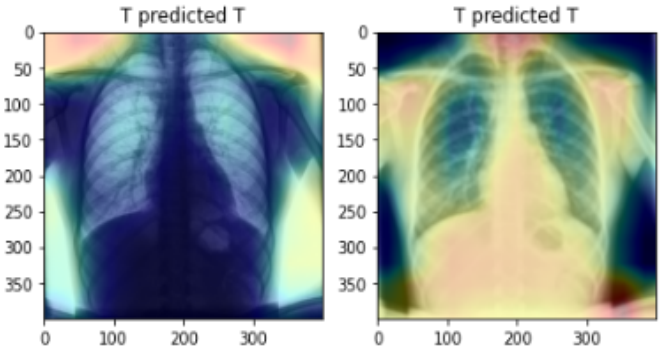
6: a lot of misclassified images are badly ruined

Class	precision	recall	f1-score
0 - Normal	0.96	0.96	0.96
1 - Pneumonia	0.98	0.96	0.97
2 - Tuberculosis	0.82	0.85	0.84

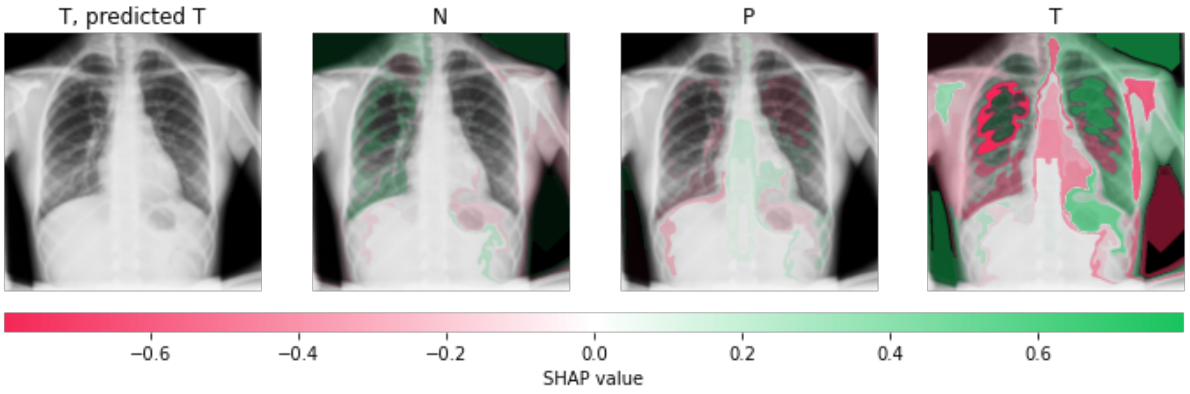


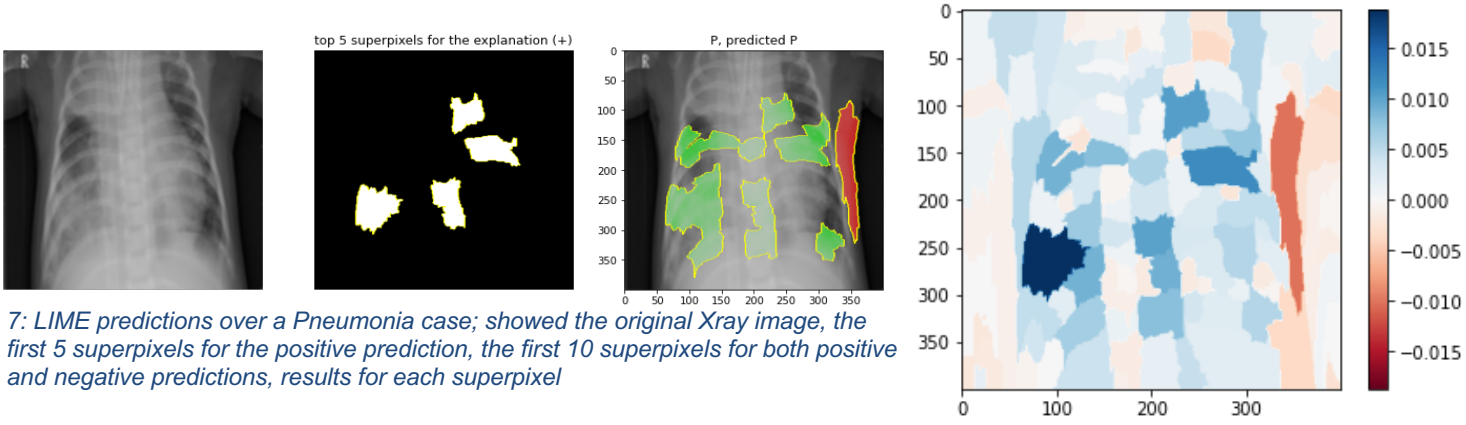
In this table are reported the results from predictions of the images in the test set. The model is less accurate in identifying TB class; this is likely due to the fact that it is underrepresented in our X-ray images, used for both training and evaluation. It is worth noticing that out of the 80 prediction errors, 24 tuberculotic patients were predicted to be “normal”. Further discussions will be done.





We also show here some of the results obtained with XAI techniques. GradCAM, in the first two figures, is not enhancing lung tissues. This may be attributed to the fact that it focuses on the last convolutional layer, preceding the GAP and the dense layers, while SHAP and LIME consider the overall features interaction.

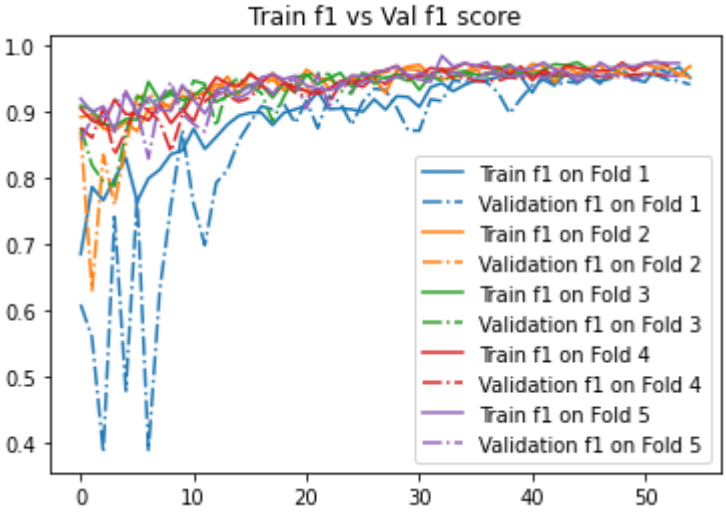




7: LIME predictions over a Pneumonia case; showed the original Xray image, the first 5 superpixels for the positive prediction, the first 10 superpixels for both positive and negative predictions, results for each superpixel

2) ResNet with dropout

Dropout is a regularization technique that involves "dropping out" or ignoring a random subset of neurons during training. In this case, the dropout rate is set to 0.2, which means that during training, 20% of the neurons in the model will be randomly ignored. This forces the model to learn multiple, independent representations of the same input, reducing overfitting and improving the performance on the test dataset.

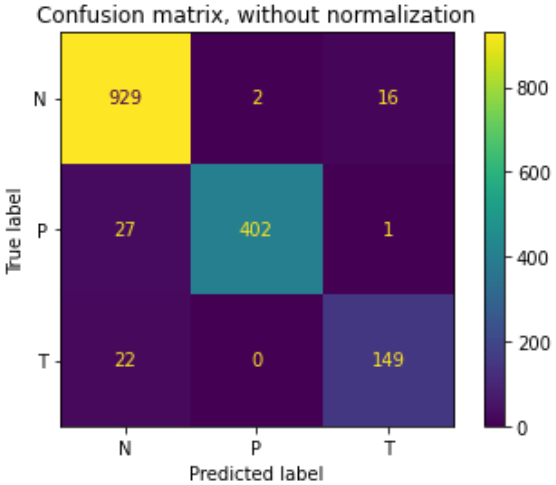


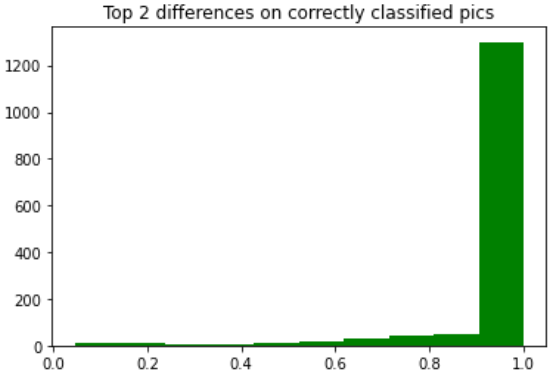
kCV results	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	wrt NO dropout
F1 score	0.9634	0.9651	0.9600	0.9629	0.9601	0.9623	+0.2%
Accuracy	0.9578	0.9642	0.9607	0.9642	0.9599	0.9614	+0.2%
Precision	0.9581	0.9644	0.9609	0.9652	0.9602	0.9618	+0.2%
Recall	0.9578	0.9635	0.9599	0.9635	0.9599	0.9609	+0.1%

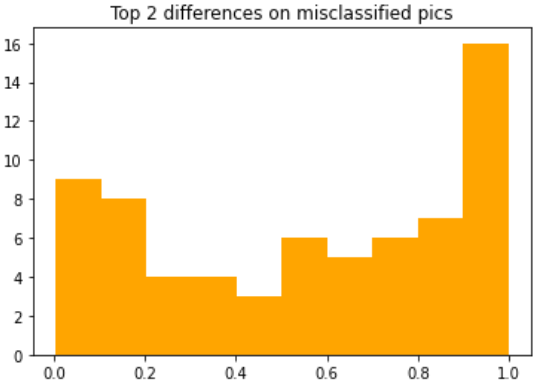
Example: 68 prediction errors out of 1548 test images (same as (1), most of the misclassified images were ruined, with various noises superimposed, and/or shifted, etc.).

Dropout regularization substantially improved scores obtained on the tuberculosis class.

class	precision	recall	f1-score
0 - Normal	0.95	0.98	0.97
1 - Pneumonia	1.00	0.93	0.96
2 - Tuberculosis	0.90 (+9.8%)	0.87 (+2.4%)	0.88 (+4.8%)



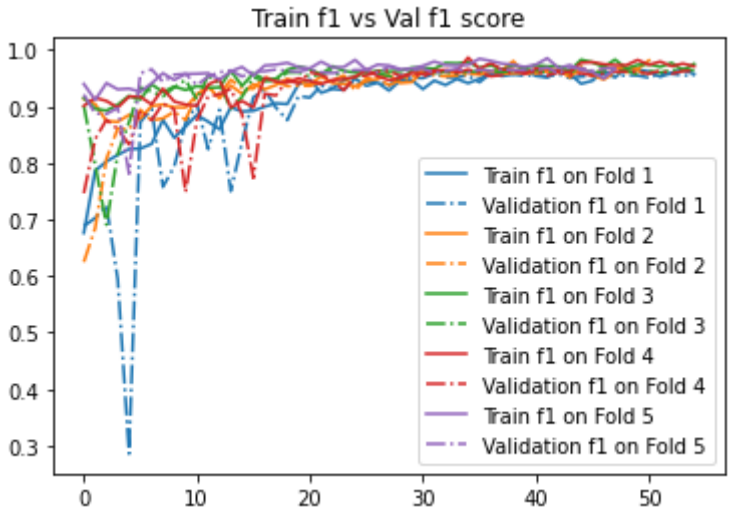




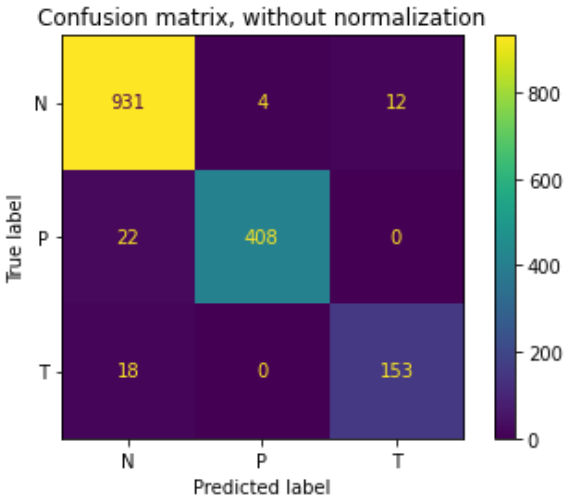
3) DenseNet

kCV results	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean
F1 score	0.9638	0.9674	0.9679	0.9658	0.9707	0.9671
Accuracy	0.9607	0.9635	0.9657	0.9649	0.9664	0.9642
Precision	0.9609	0.9625	0.9659	0.9659	0.9673	0.9645
Recall	0.9599	0.9628	0.9657	0.9649	0.9664	0.9639

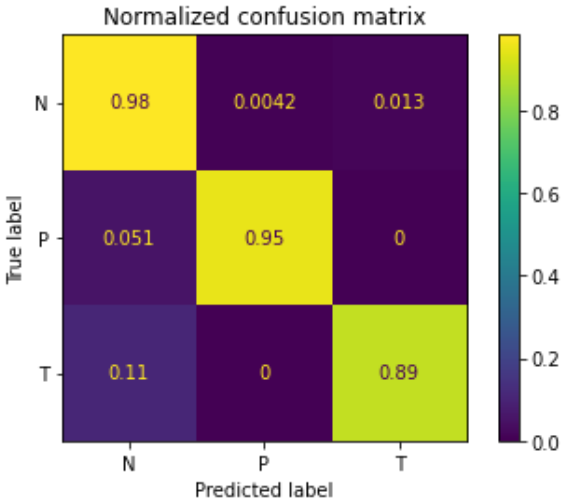
DenseNet architecure, which has way less parameters than ResNet and was not increased with fully-connected layers, shows the best results for classification. However, its performance will be discussed in the following paragraph.

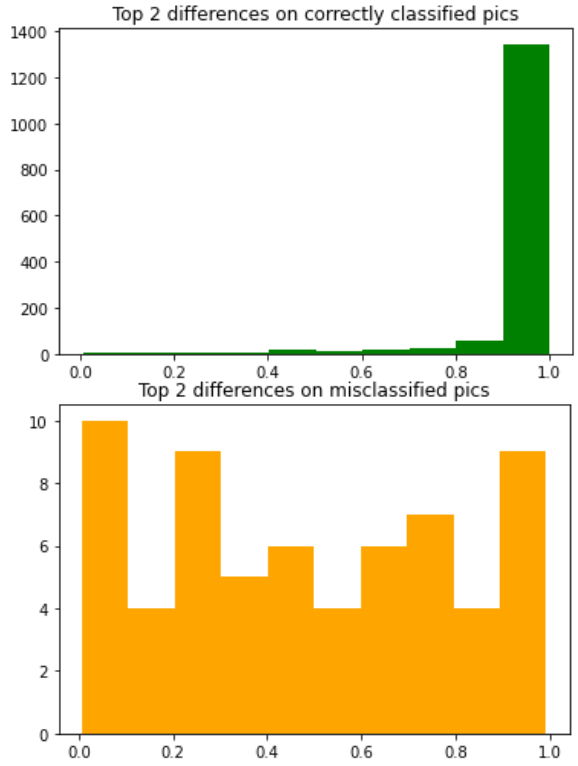


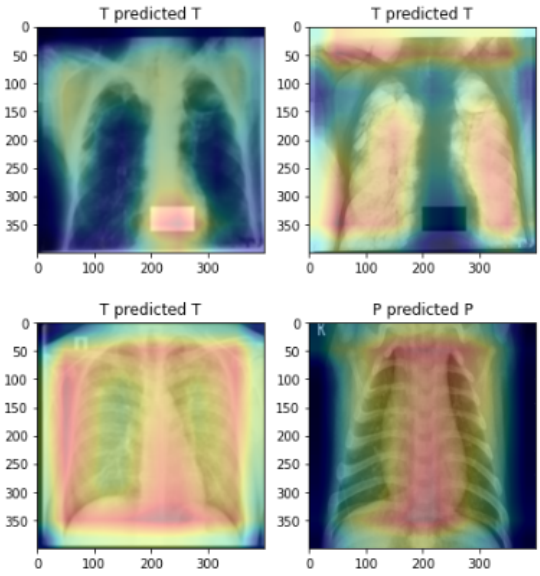
The example showed here misclassified 56 images out of 1548 hold out for the independent test. The classification report contains the percentage improvements with respect to ResNet with no dropout (1).



Class	precision	recall	f1-score
0 - Normal	0.96	0.98	0.97
1 - Pneumonia	0.99	0.95	0.97
2 - Tuberculosis	0.93 (+13.4%)	0.89 (+4.7%)	0.91 (+8.3%)

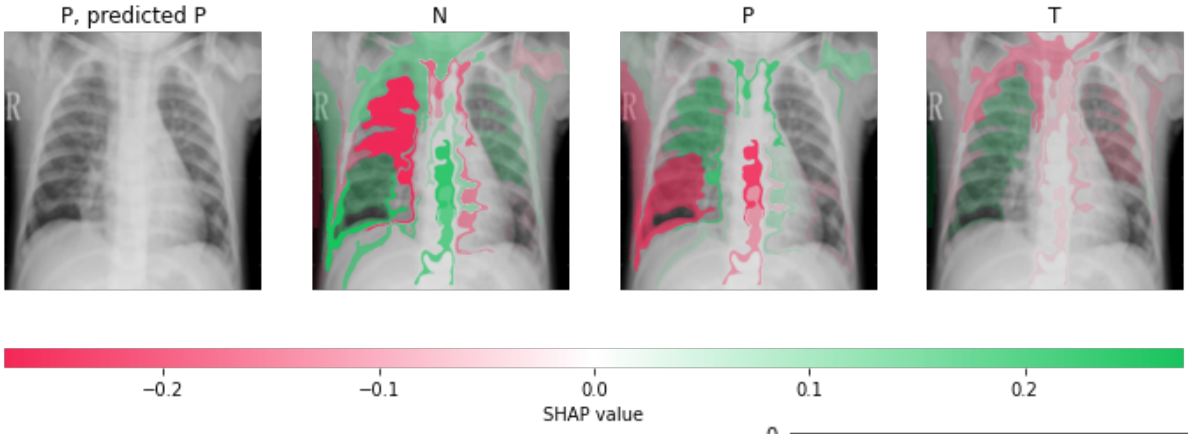


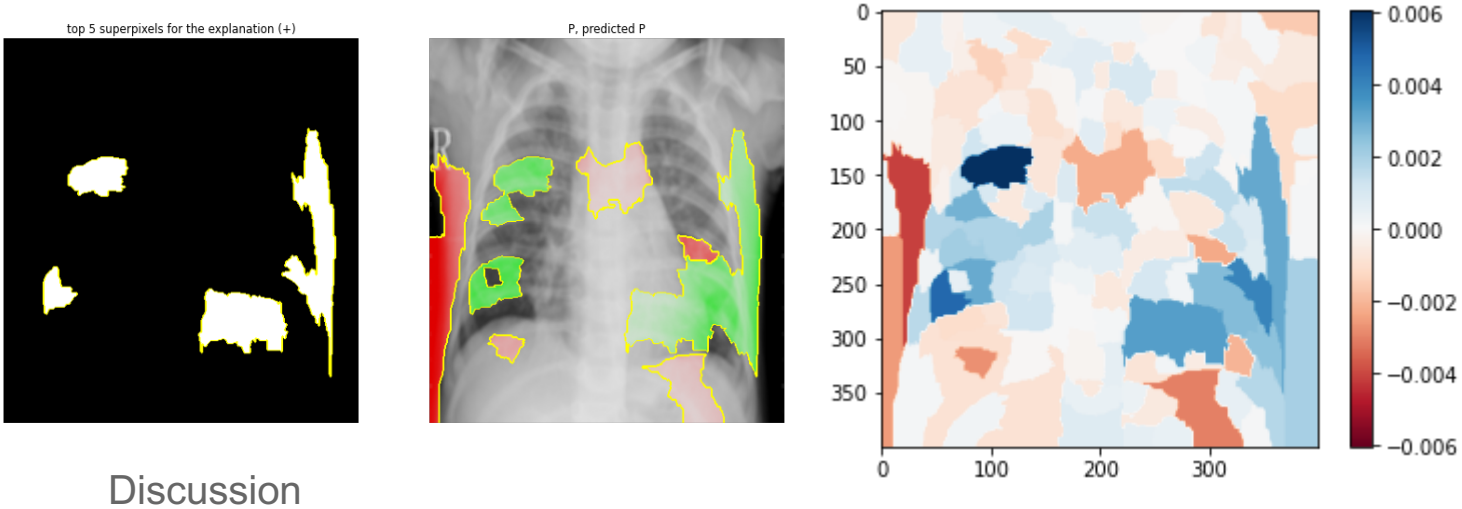




While overall performance metrics were improved compared to ResNet, the interpretability of the model's predictions was inferior. However, we now can see some results with GradCAM, which now highlights features next to the Softmax output layer. Explanation maps are not reliable on their own, and some considerations must be done. For example, the first two GradCAM heatmaps here refers to the same image, in its original version and its negative. Both were correctly predicted as "T", but in the first one the activation is pushed by the white mask, which is instead ignored in the second case.

SHAP and LIME poorly enhance lung tissues; a combination of the two may be useful, but an expert judgement is required.





Discussion

The results show that both ResNet and DenseNet models are able to achieve high performance in terms of overall accuracy and F1 score. Additionally, the use of XAI techniques such as Top-2-Differences, Uncertainty scores, Grad-CAM, SHAP and LIME helped in understanding the models' decision-making process. We usually see that XAI techniques such as SHAP and LIME are more effective than Grad-CAM. This is likely because Grad-CAM focuses on the last convolutional layer of the network, whereas SHAP and LIME provide explanations for the entire model.

Table: Results obtained on class "T"

Network	precision	recall	f1-score
ResNet50V2	0.82	0.85	0.84
ResNet50V2(∗)	0.90 (+9.8%)	0.87 (+2.4%)	0.88 (+4.8%)
DenseNet121	0.93 (+13.4%)	0.89 (+4.7%)	0.91 (+8.3%)

Both models performed well on normal and pneumonia classes, eg. DenseNet with f1 score of 0.97 on the independent test set for both classes. However, the performance on the tuberculosis class was not as high, with an F1 score of 0.84/0.88 for the ResNet model and 0.91 for the DenseNet model.

We had a limited dataset, and we must notice that the performance on unseen data may be different; also, we reshaped the images to a specific size, which could have affected the overall performances (some high-quality images were scaled down by a factor of 10). We chose to transform our grayscale images into RGB in order to exploit models pre-trained to solve a large classification problem (1000-classes classification from ImageNet) and then fine-tuned, i.e., we retrained the whole models, but with the convolutional layers initialized to the pretrained model, since enough data (>10000 labeled images) were provided. Data augmentation, i.e., our artificial attempt to increase the dataset size, can introduce biases and does not accurately reflect the variations in the real chest images. However, this technique helped us to reduce overfitting and improve the result of the explainability. As Y. Bengio wrote in “Deep Learning” (co-written with I.Goodfellow and A.Courville), in certain cases, the interpretability of the model's predictions is more important than the accuracy or F1 score, particularly in situations where the model's predictions will be used to make decisions that affect people's lives. We believe this is the case. We evaluated the proposed architectures and found that in some hyperparameters configurations and with different levels of data augmentation, while the overall accuracy was slightly below our top results, the XAI results were improved. In all our choices we always kept in mind that understanding the model’s decision-making process was essential at least as much as its f1 score obtained on the independent test set.
DenseNet121 achieves better scores, but the ResNet model seems to have better understood the task, showing us better heatmaps.
By adding dropout regularization after the GAP layer of the ResNet model, we have observed an improvement of 9.8% in precision, 2.4% in recall and 4.8% in F1-score on tuberculosis class. These results, obtained with 5-fold cross validation, indicate that the model is better able to generalize unseen data and make more accurate predictions about the presence of the infection, without improving or worsening the overall explainability. The same technique worsened DenseNet results and heatmaps, at any dropout rate.

Another limitation of these models is the trade-off between precision and recall. The results reported in the table, which were forced using a weighted loss function giving more importance to the “positive tuberculosis” cases, are an issue of great importance in machine learning and medical diagnosis. Whereas the overall performance of DenseNet seems to be better, we believe that in this case recall is more important than precision.
When studying a serious illness like tuberculosis, missing a positive case could have severe consequences: in most of our attempts, DenseNet achieved high precision scores but lower recall.

However, we want to note that a low number of false positives (high precision) prevents unnecessary treatments and follow-up tests, which also would be bad: treatments may have side effects, psychological impacts, etc.; also, they cost, they are time-consuming, and may divert resources away from patients who truly need them. That is, this procedure should be done to control a population which already has some risk of being infected, to avoid too many false positives.

The results of the study indicate that the DenseNet model achieved the best performance in terms of overall accuracy and F1 score compared to the other models tested (VGG16 andResNet). However, ResNet provided the best explanations for its decision making, with heatmaps effectively enhancing the lung tissues.
Also, it is worth noting that the ResNet model also had a much higher number of trainable parameters (25,620,611) compared to the DenseNet model (6,956,931): this means that it has more capacity to learn from the data, which allows it to achieve better performance, but also that it requires more computational resources and memory to be trained and deployed. Additionally, having more parameters also increases the risk of overfitting, meaning the model performs well on the training data but may perform poorly on unseen data.
This trade-off between performance and computational cost is something that should be considered when developing machine learning models and choosing the appropriate architecture.

Conclusion

In conclusion, this study demonstrates the potential for deep learning models to be used for the classification of tuberculosis from chest radiography images, which is in line with World Health Organization (WHO) recommendations for the use of chest radiography as a diagnostic tool for tuberculosis detection. The results of the study show that both the ResNet and DenseNet models achieved high performance in terms of overall accuracy and F1 score, with the DenseNet model achieving the best results. Also, the DenseNet model has a lower number of trainable parameters and thus required less computational resources and memory, which aligns with WHO's promotion of improving the accessibility, together with the quality, of diagnostic services. Programmatic approaches like active case finding and contact tracing involve actively searching for cases of TB in high-risk populations, which can help to identify cases that might otherwise be missed. By using XAI techniques to understand how the model arrived at a particular diagnosis, medical professionals can identify cases that might have been missed using other diagnostic methods.

References

- “Chest radiography in Tuberculosis Detection”, summary of current WHO recommendations and guidance on programmatic approaches, 2016
- Deep Residual Learning for Image Recognition, by K. He et al., 2015
- Densely Connected Convolutional Networks, by G. Huang et al., 2016
- Keras API reference https://keras.io/api/