Application of A.I. in Biomedicine – Final Assignment Report

Code available at https://github.com/RaffaeleBerzoini/AAIB_Assignment2022

Raffaele Berzoini
ID: 995498
raffaele.berzoini@mail.polimi.it

Davide Console
ID: 993506
davide.console@mail.polimi.it

Noemi Manzo
ID: 993804
noemi.manzo@mail.polimi.it

I. Introduction

There are many lung pathologies that can affect individuals, with two of the most significant being tuberculosis and pneumonia.

Tuberculosis, caused by the bacterium Mycobacterium tuberculosis, is an infection that primarily affects the lungs. It is transmitted from person to person through small droplets that are released into the air through coughing and sneezing. Common symptoms include fever, chills, night sweats, loss of appetite, weight loss, and fatigue. Nail clubbing may also occur in some cases. When a person has active TB disease, it means that the TB bacteria are multiplying and attacking the lungs or other parts of the body, such as the lymph nodes, bones, kidneys, brain, spine, and skin. From the lungs, the TB bacteria can spread through the blood or lymphatic system to other parts of the body. TB infection occurs in four stages over the course of about one month: the initial macrophage response, the growth stage, the immune control stage, and the lung cavitation stage.

Pneumonia is an infection of the lungs caused by viruses or bacteria, such as Streptococcus pneumoniae. It is an infection that causes the alveoli in the lungs to become inflamed and fill up with fluid or pus. This can make it difficult for oxygen to reach the bloodstream. Symptoms of pneumonia may include chest pain while breathing or coughing, confusion or changes in mental awareness, fatigue, fever, sweating, shaking chills, low body temperature, nausea, vomiting, or diarrhea, and shortness of breath.

These medical conditions can be diagnosed by a healthcare professional using X-ray images. In this report, we will propose and discuss various deep learning approaches for detecting these pathologies automatically using medical images.

II. Materials and Methods

A. Dataset

The dataset for this study consists of 15470 X-Ray images in PNG and JPEG format, representing a diverse range of subjects: those with pneumonia, tuberculosis, and healthy individuals. The distribution of these subjects is as follows: 60.47% (9354) normal, 27.47% (4250) pneumonia, and 12.06% (1866) tuberculosis. All images include the totality of the lungs in the coronal plane. Working with medical images, it is common to have different images for the same patient. Therefore, we assumed that all images with the same initial file name belong to the same patient and may be related to each other in some way. Therefore, by splitting the dataset, we made sure that all images for a single patient were placed in the same folder. After dividing the dataset into train and test (representing 15% of the dataset), we applied 5-layer cross-validation by dividing the train dataset. Data distribution is shown in Figure 1.

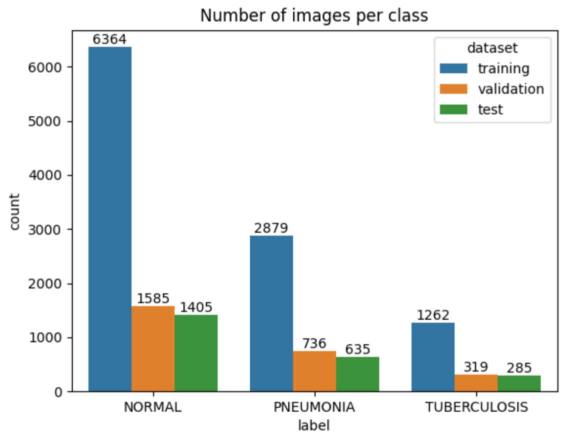


Figure 1: Distribution of images across train, validation, and test sets for each class.

B. Data Exploration

We visually inspected a selection of images from a dataset to identify any potential issue.

The dataset is very heterogeneous: some of the images exhibit characteristics such as blurriness, reversal of tissue contrast, salt and pepper noise, and uniform noise. Additionally, a small number of images are "corrupted" with the presence of black spots or have superimposed patches. Other images are padded.

To classify the images' condition, we implemented a simple algorithm that binarizes each image using Otsu's threshold [1] and counts the number of labels (i.e., the number of groups of pixels separated between one another) on the binarized image. If the number of labels is equal to or greater than 100, the image is classified as noisy/corrupted. If the image is not classified as noisy/corrupted, the algorithm extracts from the binarized image the histogram of a region of interest (ROI) comprising the four corners of the image and classifies the image as having complementary contrast if the number of 1 pixels is greater than the number of 0 pixels.

We chose to use the count of labels on the binarized image and the ROI histogram approach because it was impossible to find a single ROI suitable for all images to analyze the presence of noise automatically with SNR or histogram's metrics. Also, for this reason, we classified complementary contrast images using the four corners as ROI to have more margin of confidence even in cases where one or two corners where not suitable to be used as ROI. This classification could have been done just by looking to the histogram of the whole image (without binarization and the use of ROIs) to see if the histogram were skewed to white values or black values. However, in some cases even though the images had a normal contrast, there was a predominance of high bins (whites) (for example in some blurred images) that could have caused the failure of this method.

As shown in Figure 2, we found that approximately 60% of the dataset presented normal contrast, 20% complementary contrast, while 20% was noisy/corrupted. So, as we will see in the following section, we made sure to have all the images in the standard radiological format (normal contrast i.e., black background and white bones/tissues) and to manage noise.

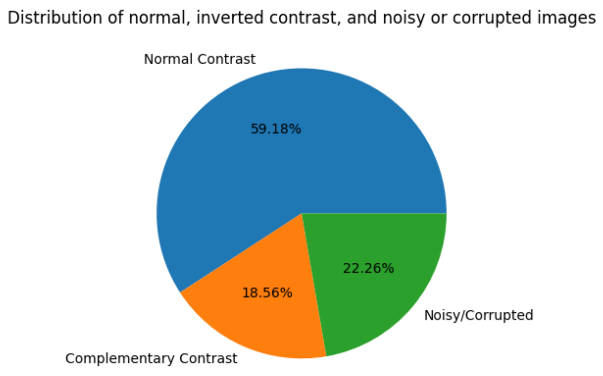


Figure 2: Percentage of normal, inverted contrast, and noisy/corrupted images in the entire dataset.

C. Preprocessing

To assess the heterogeneity, we trained the models with different preprocessing pipelines.

Our first attempt did not manage noisy and corrupted images. It just included a conversion from RGB to grayscale, a resize to the dimension 256x256, followed by the negative transformation for images not in the standard radiological format. Normalization, integrated in the first layer of the network, was always done. This basic preprocessing steps also have been used for all the following presented pipelines.

A second attempt consisted in applying a 5x5 median filter followed by a 3x3 uniform filter to all the images. These filters are very useful for removing uniform and salt and pepper noise. Nevertheless, we choose to apply the filters to all the images, as we found no particular fall in the prediction of clean images. Because these filters make the image more blurred, we tried applying an unsharp filter to better highlight the outlines. This made the image sharper but also much less contrasted, so we preferred to leave it as it was.

Another attempt was made applying data augmentation to the images. The transformations applied were a random rotation and, based on the degree of rotation, a left-right or up-and-down flip.

We trained a denoising autoencoder (DAE) [2] to remove uniform noise, since we considered this to be the most present noise and the most influent for prediction loss of accuracy. This autoencoder is made of an encoder consisting of different convolutional layers followed by two-dimensional max pooling, and a decoder consisting of convolutional layers and up sampling layers. The loss utilized was the MSE, and the network was trained to take an image with superimposed uniform noise and give as output the same image without noise, maintaining only the useful features. In case of images without noise in input, the network should be able to return an identity. Once the network was trained. It was set as the last step of the preprocessing pipeline for denoising images.

Finally, we tried preprocessing data using Principal Component Analysis (PCA) [3]. This technique is not able to eliminate noise completely, but it can be useful in reducing it. It identifies dominant patterns in the data and projects them onto a lower dimensional space, eliminating redundant or irrelevant features. After applying PCA, a subset of components is selected to be used for image reconstruction. Specifically, we applied PCA to each batch of images using a number of components equal to 12 and then reconstructed each image individually.

The comparison between these approaches is reported in detail in section III.A

D. Models

We tried two kinds of models for the classification.

We utilized different Deep Learning Neural Networks composed by a first section of convolutional layers for feature extraction, followed by a few dense layers for the classification itself. This approach is the gold standard and is the most used for image classification [4] [5]. As mentioned before, we tried different architectures, but eventually we adopted the EfficientNetB3 as it guaranteed a good tradeoff between performance and training times. To this keras model, we attached a Global Average Pooling 2D layer, a dropout layer, and a final dense layer of three neurons with *softmax* activation.

The second method we tested is not based on Deep Learning models. We trained a SVM with histogram of oriented gradients (HOG) [6]. The HOG descriptor is a feature representation method that captures the shape and structure of objects in an image. To compute it, the image is divided into blocks and blocks are divided into small cells. The intensity gradient is calculated in each cell. These gradients are then histogrammed, resulting in a compact feature vector that represents the image. HOG is robust to noise and image degradation and is able to handle variations in illumination and rotation.

E. Training

Our training allows the user to define different arguments to personalize the training of the model. Beside the standard inputs such as the number of epochs, the keras model, the training and validation sets, it takes:

- Epoch flags: the epochs at which the training should stop, update the parameters and restart. It can be both an integer, or a list of integers. For example, with epoch = 20 and epoch_flags = [5, 12, 17], the training will consist of four phases, the first from epoch 0 to 4, the second from epoch 5 to 11, and so on. If epoch_flags is an integer X, the training parameters are updated every X epochs.
- Learn rates: a list of learning rates to be used with the PolynomialDecay scheduler of Keras. As the previous argument, it can be a single float F that will be maintained throughout the whole training.
- Loss functions: a list of loss functions to use in each phase of the training. In our case we found that maintaining the categorical_crossentropy for the whole training was the best approach.
- Class weights: a dictionary (or list of dictionaries) with the weights to be assigned to each class of the dataset when computing the loss value. In our case we used the following weights for classes: Normal (N):1, Pneumonia (P): 1, Tuberculosis (T): 4.
- Adjust weights: a boolean value stating whether to update the class weight at the end of each phase of the training. With this argument set to true, the list of dictionaries of the weights are ignored from the second phase onwards. We obtained the best results with the argument set to True.
- Frozen FE: a boolean stating whether to freeze or not the training of the keras.application models in a specific case. As usual, this parameter can also be provided as a single value or a list of values.

The function automatically checks that the arguments are provided in the right format. Each phase starts with the freezing/unfreezing of the feature extractor. Then it instantiates the learning rate scheduler with the start and end learning rate provided by the user for that specific phase. It compiles the model with the loss function of the phase and then starts the training. At the end of each phase the f1-scores for each label are computed and if the adjust_wieghts parameter is set to True, they are updated based on the score obtained. The function computing the new weights uses the following formula:

$$weight_c = 1 + \left(offset - offset \times \frac{F_c^{power}}{max\left(F_{i\in[classes]}\right)^{power}} \right)$$

The higher the f1-score, the lower the weight. The higher the power and the offset, the more “aggressive” the training is towards the classes with lower f1-score. Updating the weights at the end of each phase was the approach that increased the most the homogeneity among the different classes scores.

We also implemented the possibility to load “imagenet” weights for our model (since Imagenet dataset [7] is a defacto standard for image classification) and fine-tuning [8] the model to fit our goal.

We tested with different amounts of epochs and phases for each epoch. Time requirements led us to not overcome the 180 epochs total but that were often not even reached thanks to the early stopping callback.

During the training we also applied regularization to the models. We tested both l1 and l2 regularization independently obtaining the best results using both of them with the value equal to 1e-5.

F. XAI

Explainable artificial intelligence (XAI) methods are becoming increasingly important in a variety of applications, including healthcare, finance, and transportation. XAI methods aim to provide human-understandable explanations for the decisions made by artificial intelligence (AI) systems, which can improve trust and confidence in these systems. In the healthcare industry, for example, XAI methods can help to improve the transparency and accountability of AI-powered diagnosis and treatment recommendations, which can help to ensure that these recommendations are fair, unbiased, and clinically sound.

We decided to implement different XAI methods among the ones available in literature [9].

LIME (Local Interpretable Model-Agnostic Explanations) [10] is an algorithm that can explain the predictions of any image classifier in a human-readable form, such as a heatmap highlighting the most important regions of the image that contributed to the prediction. It does this by learning a local model around the prediction made by the classifier and explaining that model rather than the original classifier itself. One advantage of LIME for image classification is that it is model-agnostic, meaning it can be used to explain the predictions of any image classifier, regardless of the type of model or the architecture of the model.

Another method we implemented is Grad-cam [11]. Grad-cam (Gradient-weighted Class Activation Mapping) is a technique for producing heatmaps highlighting the regions of an input image that are most important for making a prediction. It is often used to visualize and understand the decision-making process of a convolutional neural network (CNN). It is not model-agnostic like LIME but it can explain any CNN, regardless of the architecture or type of model.

As the third technique we implemented the occlusion sensitivity method [12]. This method is model-agnostic as LIME. The occlusion method is a technique for understanding the decision-making process of a classifier by occluding (i.e., covering or blocking out) parts of the input data and observing how the prediction changes. It is often used to visualize and understand the contribution of individual features or regions of the input data to the prediction made by the classifier.

As final techniques we decided to test an inverse method of occlusion where, instead of obscuring only part of the image, the entire image is obscured except for one part.

III. Results

A. Preprocessing

In Figure 3, we report a comparison between denoising techniques.

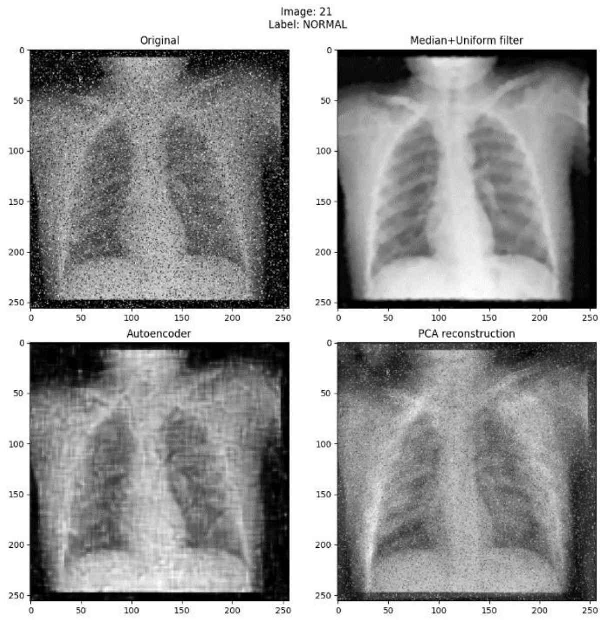


Figure 3: Adopted denoising techniques. From top to bottom, left to right: input image, median and uniform filter application, autoencoder output and PCA reconstruction.

All these techniques were tested using them in training with the same network (EfficientNetB3) and hyperparameters and same hardware resources to have a fair comparison. The performances on a single train-validation-test split were not so different and are reported in the following table.

Preprocessing	Train [%]	Validation [%]	Test [%]	Time per Epoch[†] [s]
Filtering	99.99	97.22	97.79	~160
DAE	99.99	96.42	96.57	~710
PCA	94.33	92.91	92.10	~310

[†]On a 16GB Tesla V100

Data augmentation was tried but, eventually, not used since it did not improve, nor worsen the performance.

Since the difference in performance was not significant, we decided upon using the filtering pipeline because of the reduced execution time compared to PCA (which also had a slighty worse performance) and especially to denoising autoencoders.

B. Models comparison

As already mentioned, since model performances with different denoising techniques were comparable, we decided to proceed with the five-fold cross validation with the less time-consuming model. For this reason, the definitive preprocessing adopted consisted in applying the median and uniform filtering, as well as the basic steps included in all the pipelines described in II.C.

Specifically, we performed five-fold cross validation with EfficientNetB3 and with Support Vector Classifier. In the first table we report the accuracy values for train, validation, and test splits, while in the second one we report the F1 scores for each class in the test split.

Model	ACC Train mean(std) [%]	ACC Validation mean(std) [%]	ACC Test mean(std) [%]
EfficientNetB3	99.99(0.01)	97.34(0.26)	97.45(0.29)
S. V. Classifier	99.39(0.09)	92.11(0.55)	92.27(0.28)

Model	F1 Normal mean(std) [%]	F1 Pneumonia mean(std) [%]	F1 Tuberculosis mean(std) [%]
EfficientNetB3	97.97(0.25)	98.41(0.35)	92.90(0.70)
S. V. Classifier	93.71(0.22)	94.93(0.32)	79.24(0.87)

Both the models tested with cross validation have good performances.

However, our final choice was to use EfficientNetB3. This was done not only because both accuracy and f1 scores were higher (especially on the tuberculosis class), but also because the standard deviation is overall slightly better, guaranteeing us an overall more robust model.

The robustness of the EfficientNetB3 model was also achieved by selecting the model based on validation accuracy and loss value.

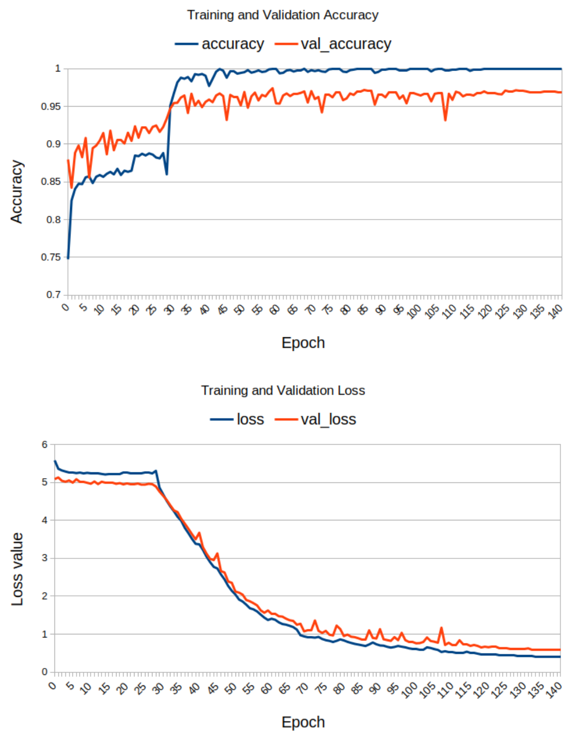


Figure 4: Training and validation accuracy and loss value for the EfficientNetB3-based model

Figure 4 shows the trend of accuracy and loss values for training and validation. There are two main points to be described in the graphs. The first one is at around epoch 30.

This is the moment where we end the last transfer learning phase, and we unfreeze the EfficientNetB3 layers to perform fine-tuning. We can see how in the first 15-25 epochs the loss has already reached a plateau. This is because we employed small fully connected layers at the end of the network that are trained very quickly with the imagenet-based features. When we unfreeze the EfficientNet we get a big increase in accuracy that made us outperform all models tested without the imagenet weights. Then, to select the best model we looked both at validation accuracy and loss value. Often the highest validation accuracy was reached in the first half of the training, however the loss was still a little bit high. While testing models with loss value greater than 1, performance on the test set were not very stable despite very similar validation accuracies. So, we always selected the model among the ones saved in the last 20-30 epochs were also the validation accuracy has reached a plateau and doesn't show anymore peaks during training. This allowed us to have multiple models of each fold that produced very similar results confirming the stability of the model at the end of the training and not a fortuitous selection of a model that accidentally performs well on our test set.

C. *Outputs of Explainability models*

After having assessed model performances and ability to discriminate among the three classes, we proceeded with the analysis on how these models provided their predictions.

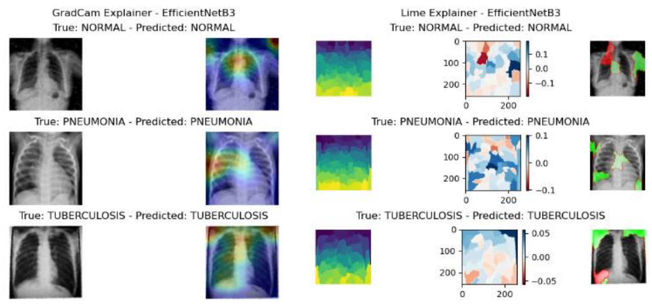

GradCam Explainer - EfficientNetB3
True: NORMAL - Predicted: NORMAL
Lime Explainer - EfficientNetB3
True: NORMAL - Predicted: NORMAL
True: PNEUMONIA - Predicted: PNEUMONIA
True: PNEUMONIA - Predicted: PNEUMONIA
True: TUBERCULOSIS - Predicted: TUBERCULOSIS
True: TUBERCULOSIS - Predicted: TUBERCULOSIS

Figure 5: EfficientNet explanations with Grad-cam and Lime approaches on test-set images. From top to bottom: normal class, pneumonia class and tuberculosis class. From left to right: Input image, Grad-cam explanation, LIME segmentation output, LIME heatmap and LIME explanation.

In Figure 5, we report some sample images of the output of the LIME and Grad-cam explainers on test-set images processed by EfficientNet. Comparison is not straightforward given the difference on the two techniques, however both explainers follow the same trends among multiple images. For tuberculosis the heatmap has higher values in the upper part of lungs lobes and on the shoulders while for pneumonia the higher values are gathered in the middle-lower part of lungs. LIME also highlights in red which areas reduced the output probability for that class.

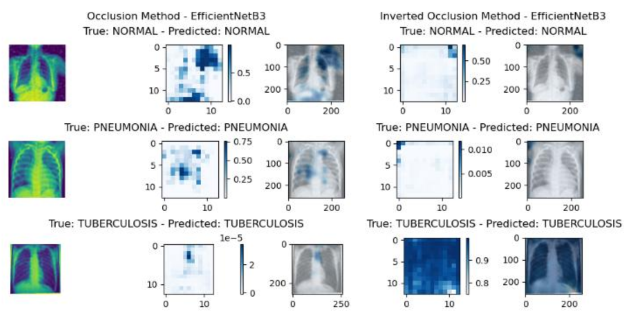

Occlusion Method - EfficientNetB3
True: NORMAL - Predicted: NORMAL
Inverted Occlusion Method - EfficientNetB3
True: NORMAL - Predicted: NORMAL
True: PNEUMONIA - Predicted: PNEUMONIA
True: PNEUMONIA - Predicted: PNEUMONIA
True: TUBERCULOSIS - Predicted: TUBERCULOSIS
True: TUBERCULOSIS - Predicted: TUBERCULOSIS

Figure 6: EfficientNet explanations with occlusion and inverted occlusion method. From top to bottom: normal class, pneumonia class and tuberculosis class. From left to right: input image, occlusion heatmap, occlusion explanation, inverted occlusion heatmap and inverted occlusion explanation.

In Figure 6, we show the results of the occlusion and inverted occlusion technique for the EfficientNet model. The occlusion method produces more consistent results with LIME and Grad-cam despite being particularly sensitive to the black patch dimensions. On the other hand, the inverted occlusion method seems unable to produce significative results with the provided image patches. The output of tuberculosis has a different explanation: our model tends to predict as tuberculosis a completely black image, making the inverted occlusion method completely ineffective.

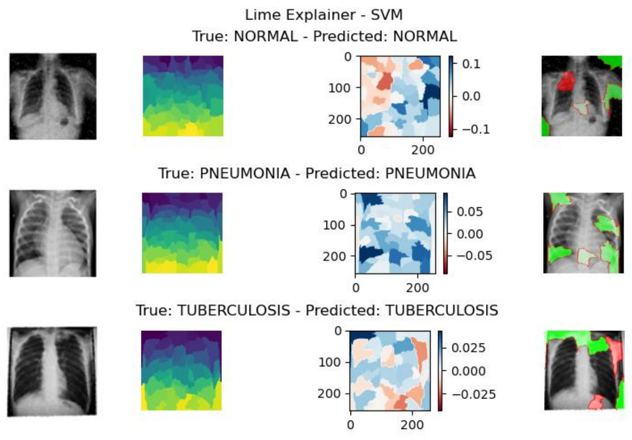
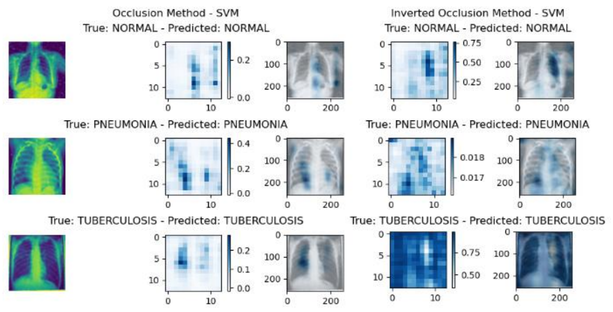

Lime Explainer - SVM
True: NORMAL - Predicted: NORMAL
True: PNEUMONIA - Predicted: PNEUMONIA
True: TUBERCULOSIS - Predicted: TUBERCULOSIS

Figure 7: SVC explanations with LIME explainer. From top to bottom: normal class, pneumonia class, and tuberculosis class. From left to right: Input image, LIME segmentation, LIME heatmap and LIME explanations.

In Figure 7, we report the explanations of the Support Vector Classifier (SVC) with LIME. Since Grad-cam is not model-agnostic and it is also model-specific [9], it cannot be applied to SVC. SVM has the same trend of LIME and Grad-cam for tuberculosis and pneumonia detection: the most important areas are at the top for tuberculosis and in the middle-lower part for pneumonia. Having similar areas with two completely different models and approaches, since the SVC has not the image as input, but it receives the histogram of oriented gradients, give us an empirical confirm that there could be a trend in the areas to look at for pneumonia and tuberculosis detection.

In Figure 8, we show the occlusion and inverted occlusion methods applied to SVC. As for EfficientNet, the occlusion method to detect pneumonia highlights areas in the lower part of the lungs. On the other hand, the tuberculosis highlights areas are a little bit lower if compared to the other explanations. Opposite to


Occlusion Method - SVM
Inverted Occlusion Method - SVM
True: NORMAL - Predicted: NORMAL
True: NORMAL - Predicted: NORMAL
True: PNEUMONIA - Predicted: PNEUMONIA
True: PNEUMONIA - Predicted: PNEUMONIA
True: TUBERCULOSIS - Predicted: TUBERCULOSIS
True: TUBERCULOSIS - Predicted: TUBERCULOSIS
Figure 8: SVC explanations with occlusion and inverted occlusion method. From top to bottom: normal class, pneumonia class and tuberculosis class. From left to right: input image, occlusion heatmap, occlusion explanation, inverted occlusion heatmap and inverted occlusion explanation.

EfficientNet the inverted occlusion method seems to be working better for normal and pneumonia classes predictions when used with SVM. With tuberculosis we encountered the same issue as before: as the CNN, also the SVC tends to classify as tuberculosis a completely black image.

The trend in using visual explanations for classifications tasks is increasing. However, different methods can lead to complementary results making the understanding of the model prediction not as easy as expected. For this reason, is always important to have, for example, a healthcare specialist to confirm or reject the obtained heatmaps.

Among the methods the most effective ones for our task and dataset seem to be Grad-cam and LIME. However, LIME has the advantage of being model-agnostic and not model-specific, making it adaptable to every classifier. Moreover, LIME highlights better also the features and areas of the input image that contributed to reducing the probability for that prediction.

IV. Conclusion

In conclusion, after having explored different solutions, we sum up the pipeline we eventually adopted.

We split our dataset in train and test set (respectively 85% and 15% of the total dataset). The train set was divided using a 5-fold cross validation. From now on we refer to the train set as the sum of the data in the 5-1 folds.

The images were preprocessed with the following steps:

- They were loaded in grayscale
- They were resized to 256x256
- Images not in the standard radiological format were adjusted applying a negative transformation
- A 5x5 median filter was applied to remove salt and pepper noise
- A 3x3 uniform filter was applied to manage images with uniform noise
- Images were converted from grayscale to RGB format to be able to use "imagenet" weights during training.

We loaded 32 images per batch. We trained EfficientNetB3 loading "imagenet" weights for 180 epochs with the following initial class weights: N: 1, P: 1, T: 4. For the first 30 epochs we trained only the last layers, freezing EfficientNetB3. From the 30th epoch onwards we fine-tuned the entire network. At each epoch, the learning rate was reduced using a polynomial decay. Furthermore, every 20 epochs the class weights were updated to encourage better performance on the worst performing class. The hyperparameters we used for the weights' computation are *power* equal to 5 and *offset* equal to 3. To improve generalization and avoid overfitting, we applied l1 and l2 regularization (l1=l2=1e-5) as well as dropout (dropout=0.4).

After five-fold cross validation, we tested each one of the five models on the test set, and finally choose the one with the best performance for the inference phase.

We analyzed the results with four different techniques of Explainability, finding some patterns among them. Inverted occlusion is the least effective one, while LIME and Grad-cam are the most consistent among each other.

V. References

[1] N. Otsu, "A threshold selection method from gray-level histograms.," *IEEE transactions on systems, man, and cybernetics,* vol. 9, no. 1, pp. 62-66, 1979.

[2] L. Gondara, ""Medical image denoising using convolutional denoising autoencoders."," *IEEE 16th international conference on data mining workshops (ICDMW),* pp. 241-246, 2016.

[3] P. C. A. B. José V. Manjón, "MRI noise estimation and denoising using non-local PCA.," *Medical Image Analysis,* vol. 22, no. 1, pp. 35-47, 2015.

[4] T. Rahman, M. E. Chowdhury, A. Khandakar, K. R. Islam, K. F. Islam, Z. B. Mahbub and e. al., "Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray," 2020.

[5] Q. Guan and Y. Huang, "Multi-label chest X-ray image classification via category-wise residual attention learning," 2020.

[6] S. Gornale, U. Pooja, S. Kiran and S. Prakash, "Determination of osteoarthritis using histogram of oriented gradients and multiclass SVM.," *International Journal of Image, Graphics and Signal Processing,* vol. 9, no. 12, p. 41, 2017.

[7] J. Deng, W. Dong, R. Socher, L. J. Li, K. L. and L. Fei-Fei, ""ImageNet: A large-scale hierarchical image database"," *2009 IEEE Conference on Computer Vision and Pattern Recognition,* pp. pp. 248-255, 2009.

[8] N. T. e. al., ""Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?"," *IEEE Transactions on Medical Imaging,* vol. 35, no. 5, pp. 1299-1312, 2016.

[9] B. H. van der Velden, H. J. Kuijf, K. G. Gilhuijs and M. A. Viergever, "Explainable artificial intelligence (XAI) in deep learning-based medical image analysis.," in *Medical Image Analysis,* 2022.

[10] M. T. Ribeiro, S. Singh and C. Guestrin, "" Why should i trust you?" Explaining the predictions of any classifier.," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization.," in *Proceedings of the IEEE international conference on computer vision*, 2017.

[12] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, 2014.