Lung Diseases Classification from CRX images using Machine Learning and Deep Learning

1st Saigol Ahmad
M.Sc. Mechatronics
Hamburg University of Technology
Hamburg, Germany
ahmadsaigol1@gmail.com

2nd Romero Joel
M.Sc. Mechatronics
Hamburg University of Technology
Hamburg, Germany
joel.romero.munoz@tuhh.de

***Abstract*—COVID-19 continues to be a global health affair. Early detection through CRX Images is an important counter-measure. It plays a key role in containing COVID-19 spread, which among other methods can be done using CRX images. Furthermore, other diseases such as pneumonia, present similar symptoms and can be also detected through the mentioned method. In this study, we built systems that can detect COVID-19 in CRX images, as well as other diseases, using classical Machine Learning and Deep Learning methods and an empirical comparison was carried out. It was found that Haralick and Zernike feature extractors, together with Support Vector Machine as classifier, obtained the highest balanced accuracy within an ML approach: 88.03% for binary and 82.01% for multiclass classification. The pretrained neuronal network MobileNet V2 provided the highest balanced accuracy of 90.67% for binary and 87.59% for multiclass classification.**

***Index Terms*—Covid-19, Deep Learning, Machine Learning**

I. Introduction

A. Background and motivation

On March 2020, the COVID-19 outbreak was declared a pandemic. Ever since, vaccines have been developed, but their efficacy is still 95% in terms of preventive action, as it contributes to immunity via the production of antibodies, and are not even a cure per se. Therefore, early diagnosis of COVID-19 is still important, as it can be traced to avoid further spread. To accomplish this, one methodology used by healthcare professionals is to analyze computed tomography (CT) and chest X-ray (CXR) images. Currently, they are focusing efforts on early detection of coronavirus: only in the first 80 days of 2020, about 1245 academic articles were written. Deep Learning (DL) is an State of the Art method for COVID-19 detection using CXR and CT Images [1].

B. Problem overview

This study aims to perform COVID-19 detection and differentiate it from other respiratory conditions, by contrasting or combining feature extraction with traditional machine learning methods, and by using deep learning. A specific CXR database was provided. Two types of classification were required: Binary and Multiclass. Binary Classification would determine whether a patient had Covid or not. Multiclass Classification would diagnose 4 different conditions: Healthy (Normal) lungs, Covid-19, Pneumonia and Lung Opacity.

C. Objectives

This study was divided into two phases. During Phase 1, the use of feature extractors and supervised machine learning models was demanded for Classification. During Phase 2, the use of Deep Learning models was required. In the end, the major objectives are:

- o Evaluate how preprocessing affects the performance of the classification models.
- o Evaluate the effect of feature extractors hyperparameters on overall model performance.
- o Determine an appropriate Classifier for ML models.
- o Analyze the impact of Augmentation in DL models.
- o Contrast the performance of ML models against DL models, plus the advantages and disadvantages of using each of them.

II. Related Works

Several studies are being conducted to find artificial intelligence solutions for respiratory diseases detection, and specially for COVID-19 due to the current worldwide health status.

Emre A. [2] conducted a research using ML, which consisted of applying segmentation to CRX images, to extract only the area of interest by masking the images with a circle that shares the same center as the chest. Posteriorly, the images were resized to 15x15 pixels and taken to the RGB color space. Each one of the 255 pixels' intensity value were a used as a feature vector. The classifier Support Vector Machine (SVM) reached 93% of balanced accuracy on the test success. Juliana Gomes et al. [3] developed an intelligent tool for COVID detection without segmentation. In this case, Haralick and Zernike moments were instead used as feature extractors. In the end, SVM obtained a balanced accuracy of 89.78%.

Geeta Rani et al. [4] performed an extensive study with a DL approach. Preprocessing was a step were researchers put a lot of emphasis. Firstly, the dataset was analyzed by radiologists to separate bad quality CRX images, which are produced by a lack of respiratory capacity in sick patients and can undermine the performance of the model. Then, bony structures such as neck, arms, ribs, collar bones, and solid tissues outside the lung region were removed as they interfere in the feature extraction process. To do this, a bone shadow suppression model and a lung segmentation model were applied, and as a result, only the lung region was extracted from the CRX. Finally, the extracted lung areas were used as input for an EXP-Net model. They determined that lung segmentation increased the DL classifiers' overall accuracy by an average of 2.7% and bone suppression increased it by 4%. Also, EXP-Net achieved an accuracy of 94.07%. Dongguan Li et al. [5] developed a DL high accuracy platform to detect Covid-19 from CRX images. The core approach of the study was to generate a voting algorithm that combined 17 pretrained CNNs (including ResNet18, RestNet50, MobileNetv2), to overcome image variations. Five different binary classifiers were used to assign the images into 4 categories: Healthy, Bacteria, Covid-19, and other virus. An accuracy of 99.62% was obtained when averaging the performance of all the classifiers, i.e., the final accuracy of the proposed model.

III. Materials and Methods

A. *Dataset description*

To perform this study a dataset of CXR images were provided, which included 16930 images for training along with its labels, 4235 images, each with and without noise, without labels for testing. The training dataset was imbalanced, i.e., there were 17.1% images of COVID-19, 28.4% images of Lung Opacity, 48.16% images of normal and 6.34% images of pneumonia [6].

B. *Data Splitting*

To validate our models, the images of the training set were split into training and validation subsets. Random generation of subsets is not always the most appropriate method, as the distribution of the target variables can differ between datasets. Moreover, in the case of class imbalance within the training dataset, where instances of the target variables are not evenly distributed, intended model's performance can be reduced even more. This was the case for the provided dataset, as previously mentioned, where the number of instances for the class "Normal" exceeds by four times the number for "COVID". Hence, the Stratified Sampling technique was applied during data splitting. It consists of forcing the distribution of instances of the target variable(s) to be the same among different subsets [7]. The raw data was then split into 70% for the training subset and 30% for the validation subset.

C. *Data Preprocessing*

To effectively process the information, different operations where performed to the images within the training subset. These operations mentioned bellow were tested in different combinations.

Normalization: It was used to map the images from [0-255] to [0-1] so values obtained from feature extractors can be comparable, except when working with Haralick and Zernike Features. For normalization of features, Standard Scaler normalization was used. It involves transforming the values of a dataset so that they have a mean of 0 and a standard deviation of 1. In all cases within this study, except for pretrained Neuronal Networks (NNs), these values were calculated during training and used during evaluation. For pretrained NNs, values found in referenced researches were used.

RGB transformation: Pretrained NNs require images with three color channels, while the images of the proportioned dataset were gray. To solve this, it is a common practice to perform a gray to RGB transformation, which will copy the same information of the gray image in three different channels, as performed by Emre A. [2].

Resizing: Images of the same dataset were converted to a determined same size, so ML algorithms could handle them in a consistent manner. Additionally, pretrained Neuronal Networks expect an input of specific size. Thus, images were resized to match the required shape.

Bilateral Filter: Haralick and Zernike Features are susceptible to noise, so it was necessary to denoise the image before extracting these features. One way to denoise image is the use of bilateral filter. It performs a pixelwise operation that uses the neighborhood pixels to determine the spatial and intensity distance between them. It was used as it reduces any existent noise within the image, while preserving the texture of the image, which is decisive for the problem being tackled in this research [8].

Augmentation: Upon investigating the noisy dataset, several aspects were identified which were likely to make the model perform worse on them. Hence, Augmentation was applied on the training dataset to make models robust to noise. It consists of applying different operations to existent images and storing the results. Additionally, Augmentation is usually used when there is an insufficient volume of data to train the model [9], which was the case for the CNNs' training of the present study, as it provided noisy images to the training dataset. Applied operations included: translation, rotation, affine transformation (translation by 10-30%, and rotating between -15 and 15 degree), horizontal flip, vertical flip, and cropping (patch of 224x224 and then resized to match the size of original image). Online and Offline Augmentation were applied to train the Deep Learning models. Online consisted of applying Augmentation to the input of the model, i.e., all the images of the training subset, while training it, by randomly applying the operations previously mentioned individually or as a combination of them. Offline consisted of generating additional images by augmenting the training dataset, preserving the tags from the original images, and adding the results to indicated dataset prior to starting to train the model. Total of 4235 images were added to the training dataset. Out of these, 80% consisted only of single transformation while remaining 20% consisted of images upon which random number of random transformations were applied. Furthermore, these images were generated in such a way that the original distribution of classes in the dataset was maintained.

D. *Feature Extractors*

During Phase 1, Machine Learning models required an input with information summarized from images. Feature extractors were used to obtain this representative information.

Histogram: It is a way to quantify the intensity of pixels and cluster them groups representing ranges, called bins. As a result, a vector of accumulated intensities was obtained. It was used as it is a more quantifiable representation of a images.

Skewness: It is a measure of asymmetry of an image's intensity distribution given the center as a reference point. It can be measured with the Fisher-Pearson coefficient as indicated in (1):

$$g_1 = \frac{\sum_{i=1}^{N}(Y_i - \overline{Y})/N}{s^3} \tag{1}$$

where $\overline{Y}$ is the mean intensity, s is the standard deviation, and N is the number of pixels.

Kurtosis: It is a measure of whether the data is heavy tailed or light-tailed distributed compared to a normal distribution. The formal definition is shown in (2) [10]:

$$kurtosis = \frac{\sum_{i=1}^{N}\left(Y_i - \overline{Y}\right)^4/N}{s^4} \tag{2}$$

Haralick: Radiologists visually segregate different surfaces and its textures within the lungs to assess their condition, and whether there are uncommon conditions. Texture, an intrinsic property of a surface, contains important information about its structural composition. Therefore, feature extractors that can analyze such patterns, provide significant information for the stated problem. Haralick moments determine different patterns and their location throughout an image, by calculating statistical information associated to co-occurrence matrices, as shown in Fig. 1. The elements p(i,j) of these matrices represent the probability of going from one pixel with intensity i to another pixel of intensity j, according to a certain distance and an angle of the neighborhood. By doing this, spatial distribution and levels of gray are obtained in all four directions. By taking the average of these directions and calculating the 13 statistical measurements (features) over it, rotationally invariant features are obtained, as done by Juliana Gomes et al. [3].

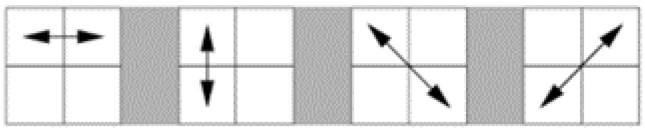


Fig. 1. Different levels of proximity for a pixel [11].

Hyperparameters used to tune this feature extractor were Blur and Distance. Blur smooths the image before the feature extraction. Distance determines the size of the neighborhood, by setting the maximum distance between two pixels that are considered neighbors.

Zernike: It captures information related to shape or geometry from an image. Zernike moments are resistant to rotation, non-redundant, and tolerant to noise. These moments are computed from the image's intensity function projections on the orthogonal base functions. To do so, the image's center is considered as the center of a unit disk, as shown in Fig. 2. Each of the moments are then calculated from the equations (3) and (4):

$$V_{n,m}(\rho,\theta) = R_{n,m}(\rho)e^{-jm\theta} \tag{3}$$

$$R_{n,m}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!}\rho^{n-2s} \tag{4}$$

where $V_{n,m}$ will be the resultant moment, from the polynomial order n and a given parameter m.

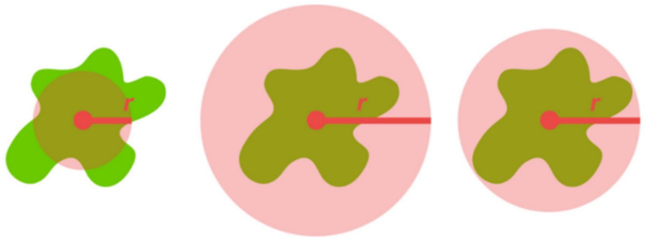


Fig. 2. Radius size surrounding area of interest [12].

Hyperparameters used to tune this feature extractor were Degrees and Radius. Degrees determines the maximum degree of the Zernike moments that are calculated, as a result, the total number of them. Radius determines the size of the circle used to sample the image intensity values for computing the Zernike moments.

E. *Phase 1: ML-based Classification*

Once having extracted features, they were used with different classifiers to build models and perform classification. The ones considered for this study are mentioned below.

Support Vector Machine: SVM creates hyperplanes, which partition the data to classify it. In SVM, the radial basis function (RBF) kernel is a function used to separate data into distinct classes. The RBF kernel is a non-linear kernel that allows the SVM to handle non-linearly separable data classes by transforming the data into a higher-dimensional feature space, where it becomes more separable. It should be noted that SVM is a ML classifier that has shown a significant performance in similar classification studies previously performed, which is the reason it was also used in this study.

Random Forest: Also called Decision Tree Forests, combine bagging (generation of datasets by sampling original training data) and random feature selection to add diversity to the decision tree models. After trees are generated, the model uses a vote to combine trees' predictions. It was chosen as an alternative model as they can handle large datasets, as only a small portion of the full feature set is used.

Boosting: It boosts performance of weak learners using models trained on resampled data and a weighted vote to determine the final prediction. Contrary to bagging, resampled datasets are constructed specifically to generate complementary learners. A common boosting algorithm is AdaBoost or adaptative boosting, which is tree-based implemented. This model can also effectively handle large datasets, and it was chosen for this reason.

F. Phase 2: NN-based Classification

Posteriorly, Neuronal Networks (NNs) were an additional tool used, to achieve a better performance or as an alternative classification method.

NNs use artificial neurons, shown in Fig. 3, as building blocks to model complex data. The neuron is modeled mathematically as indicated by (5):

$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{5}$$

where x_i represents the value of a feature i, w_i the weight given to each feature, and f is the activation function. A first strategy included training and using a simple NN as a classifier, with its inputs being the results obtained from feature extractors mentioned in Phase 1. We tested with different configurations of hyperparameters (number of neurons=8,16,32,64 number of layers: 1,2,3), and we found that 32 neurons with a single hidden layer performed the best. The activation function used for Haralick and Zernike feature extractors was Leaky ReLU, with parameters alpha=0.01 and lambda=0.01. Convolutional Neuronal Networks (CNNs) are

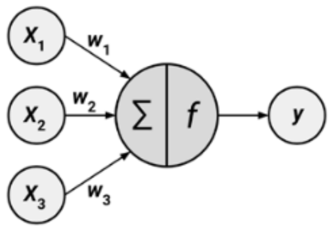
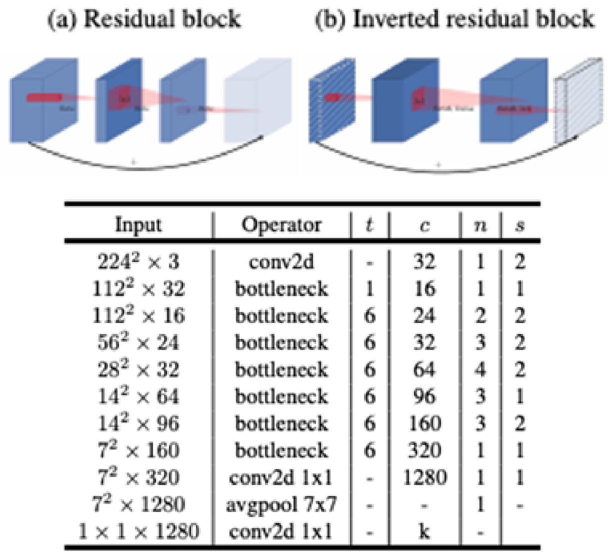


Fig. 3. Artificial neuron.

NNs that include at least one convolutional layer. A typical CNN consists of the combination of convolutional layers, pooling layers, and dense layers. A second strategy within this study was training a CNN with preprocessed images and posteriorly using it as a classifier. The architecture of this CNN was arbitrarily generated. Our model looked as follows: Conv (32 filters, size=7) – LReLU – MaxPool – Conv (64 filters, size=5) – LReLU – MaxPool – Conv (128 filters, size=3) – LReLU – MaxPool – Flatten – Dense (256 neurons) – LReLU – Dense (128 neurons) – LReLU – Output. All convolutional layers used stride=1, activation function a value of alpha=0.01, and pooling layer with stride=2.

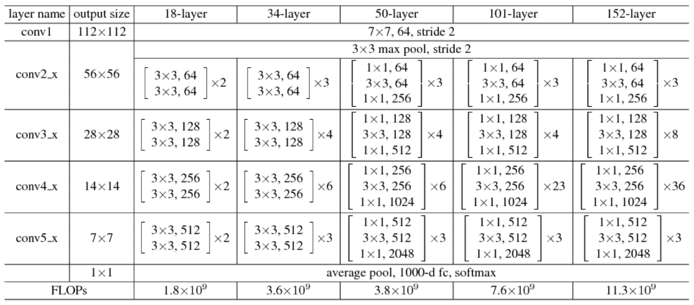
Pre-trained CNNs have been previously trained for other classification tasks, and have already learnt pattens and features, which can be reused for a different problem. When doing so, the last layer is replaced by a layer with number of neurons equal to number of classes, while the weights obtained in the rest of the layers remain the same. For the scope of this study, a last strategy consisted of individually testing three pre-trained CNNs mentioned in the Literature: MobileNet V2 shown in Fig. 4, Resnet18 and Resnet 50 shown in Fig. 5.



(a) Residual block
(b) Inverted residual block

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fig. 4. MobileNet V2 structure [13].



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 5. Resnet 18 and Resnet 50 structure [14].

G. Evaluation Metrics

In order to measure the performance of different models, the metric balanced accuracy was used. The concepts related to this metric are described below.

Confusion matrix: It is a table that categorizes predictions according to whether they belong or not to a class. The size of the matrix depends on the number of classes. The class of interest is known as positive class, and the others are negative. The matrix will be composed of the following categories, as shown in Fig. 6 for a two classes example:

- o True Positive (TP): Correctly classified as class of interest.
- o True Negative (TN): Correctly classified as not class of interest.
- o False Positive (FP): Incorrectly classified as class of interest.

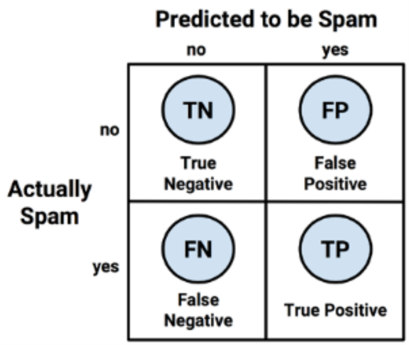


Fig. 6. Two-class confusion matrix [15].

- o False Negative (FN): Incorrectly classified as not class of interest.

The importance of this matrix falls in all the performance metrics derived from it.

Balanced accuracy: Balanced accuracy is useful used when the dataset is imbalanced. It is defined as indicated by (6) :

$$Balanced\ Accuracy = \frac{1}{2}\left(\frac{\text{TP}}{TP+FN} + \frac{\text{TN}}{TN+FP}\right) \quad (6)$$

H. Loss function and Optimizer

A loss function and an optimizer are required to handle the process of updating and choosing the best possible parameters for the classification model.

A loss function measures the difference between predicted and true probability distributions. Weighted cross entropy loss is a variation of cross entropy loss. It is especially useful for class imbalance, as it assigns different weights to different classes, as shown in (7) [16]:

$$WCE = -\sum_{i=0}^{n} \alpha_{\text{i}}\, y_i \log{(p_i)} \quad (7)$$

where, α_{i} is the weight assigned to the class i, y_i the true label (either 1 or 0), p_i is the probability of the class i, and n is the total number of classes. For the scope of this study, weights were calculated using (8):

$$\alpha_{\text{i}} = 1 - \frac{\text{number of images of class i}}{\text{total number of images}} \quad (8)$$

An optimizer is required to update the weights of a neuronal network. Adam optimizer avoids oscillations and convergence problems during training. It can be tuned with various parameters [17]. The following values for the hyperparameters of Adam were chosen Betas β_1 =0.9, β_2 =0.999, and Learning rate lr=0.001. The methodology of this study is summarized in Table I.

IV. Results and Discussion

A. Phase 1

SVM obtained a higher balanced accuracy when compared to other ML algorithms, as show in Table II. The comparison was performed using Haralick and Zernike moments as

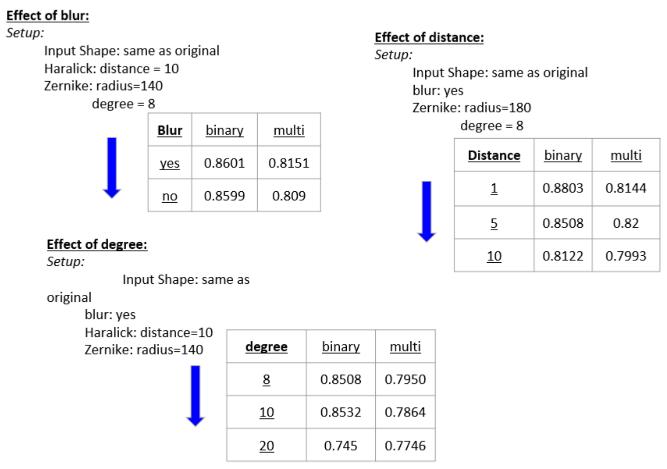
TABLE I
Methodology of Lung diseases classification study

Test	Phase	Feature extraction	Classifier
1	1	Histogram, Contrast, RMS, Skewness, Kurtosis, Haralick, Zernike, and possible combinations	SVM, RFT, Boosting
2	2	Haralick and Zernike	Neuronal Network
3	2	Convolutional Neuronal Network	
4	2	Pre-trained NN: MobileNet V2	
5	2	Pre-trained NN: ResNet18	
6	2	Pre-trained NN: ResNet50	

feature extractors, and hyperparameters values Distance=1, Radius=180 and Degree=8. A difference of 16.89% for binary and 11.04% for multiclass classification, compared to Boosting. The values obtained were superior by 16.89% for binary and 11.61% for multiclass classification, compared to Boosting. Therefore, SVM was chosen for further work and comparison between feature extractors performance.

The relation between the model accuracy and the feature extractors with SVM as classifier is shown in Table III. Haralick and Zernike moments together outperform the rest of combinations tested, with a balanced accuracy of 88.03% for binary and 82.01% for multiclass classification. When both feature extractors work independently, the accuracy can reduce as much as 12% in the case of Zernike and binary classification.

Haralick and Zernike hyperparameters were optimized, as shown in Fig. 7. Variation of Blur, either including it or not, shows that including the total accuracy up to 0.6% for multiclass classification. Variation of Distance shows that a smaller distance will increase the total accuracy in up to 6.81% for binary and 1.51% for multiclass classification, when comparing values of 1 and 10. Variation of Degree shows that a smaller distance will increase the total accuracy in up to 10.58% for binary and 2.04% for multiclass classification, when comparing values of 8 and 20.



Effect of blur:
Setup:
Input Shape: same as original
Haralick: distance = 10
Zernike: radius=140
degree = 8

Blur	binary	multi
yes	0.8601	0.8151
no	0.8599	0.809

Effect of distance:
Setup:
Input Shape: same as original
blur: yes
Zernike: radius=180
degree = 8

Distance	binary	multi
1	0.8803	0.8144
5	0.8508	0.82
10	0.8122	0.7993

Effect of degree:
Setup:
Input Shape: same as original
blur: yes
Haralick: distance=10
Zernike: radius=140

degree	binary	multi
8	0.8508	0.7950
10	0.8532	0.7864
20	0.745	0.7746

Fig. 7. Hyperparameters variation on Haralick and Zernike.

B. Phase 2

Using an NN as a classifier, with its inputs being the feature extractors of Phase 1, Haralick and Zernike moments, produced slightly better results in terms of accuracy than SVM. A difference of 2.81% for binary and 0.52% for multiclass classification, compared to SVM, was obtained as shown in Table IV. This can be explained due to the obtainable nonlinearity in NN.

Results using different DL models are also shown in Table IV. A simple CNN, without Augmentation in the preprocessing phase, performed the best for binary classification in clean datasets with 92.75% of balanced accuracy, while MobileNet V2 did in multiclass classification with 88.92%. It can be observed that the randomness of Online Augmentation decreased the general performance of every model for clean datasets. ResNet50 provided the highest balanced accuracy of 85.21% for binary and 81.95% for multiclass classification. Offline Augmentation produces higher accuracy in every model when compared to Offline Augmentation. MobileNet V2 provided the highest balanced accuracy of 90.67% for binary and 87.59% for multiclass classification.

In the case of noisy images, Table V show the balanced accuracy obtained with different NNs. A simple CNN outperformed in binary classification with 84.39% of balanced accuracy. A simple NN, with Haralick and Zernike moments as feature extractors, outperformed in multiclass classification with 78.80% of balanced accuracy.

TABLE II
BALANCED ACCURACY COMPARISON FOR ML CLASSIFIERS

Classifiers	Settings	Binary Classification	Multiclass Classification
SVM	C_SVC, RBF, hyperparameters (k-fold cross validation)	0.8803	0.8144
RFT	MaxDepth=20	0.7517	
Boosting	Number of weak classifiers = 10, MaxDepth=20	0.7114	
5	2	Pre-trained NN: ResNet18	
6	2	Pre-trained NN: ResNet50	

V. CONCLUSIONS & RECOMMENDATIONS

Haralick and Zernike, together with SVM, performed the best for the objective of this study. Better performance can be achieved for Haralick and Zernike by using Blur as well as small values for the hyperparameters Distance and Degree.

The pretrained CNN MobileNet V2, combined with Online Augmentation, provided better results that the models tested with Offline Augmentation. These results could be improved even further by training the model for long time, especially when using Online Augmentation, as this technique becomes effective for the model in handling noise by doing so. A simple CNN together with a NN performed the best for a noisy dataset.

TABLE III
BALANCED ACCURACY RELATED TO FEATURE EXTRACTION

Features	Settings	Classification	Validation Balanced Accuracy
Basic Extractors		Binary	0.6303
		Multi	0.4242
Histogram	256 bins	Binary	0.8361
		Multi	0.7306
Haralick	distance=10, blur	Binary	0.817
		Multi	0.685
Zernike	blur, radius=180	Binary	0.721
		Multi	0.741
Haralick and Zernike	blur, distance=1, radius=180, degree=8	Binary	0.8803
	blur, distance=1, radius=140, degree=8	Multi	0.8201
All		Binary	0.8438
		Multi	0.734

The present study only considered the last layer training of pre-trained CNNs. Therefore, training more layers can possibly improve the results, specifically when it comes to handling a noisy data.

Based on the literature review, quality of the CRX images of the training database must be observed to develop more accurate DL detection models. Also, additional steps such as lung segmentation or combination of pretrained NNs would allow more precise results.

Finally, the balanced accuracy obtained with the best models in this study are approximate to ones obtained in similar researches, with a difference no bigger than 10%.

TABLE IV
BALANCED ACCURACY COMPARISON FOR DL CLASSIFIERS

Models	Without Augmentation		Online Augmentation		Offline Augmentation	
	Binary	*Multi*	*Binary*	*Multi*	*Binary*	*Multi*
NN	0.908	0.825	0.84	0.78	0.87	0.811
CNN	0.928	0.871	0.794	0.451	0.879	0.854
MobileNet	0.929	0.889	0.829	0.778	0.907	0.876
ResNet18	0.891	0.853	0.79	0.76	0.859	0.832
ResNet50	0.918	0.879	0.852	0.82	0.902	0.865

TABLE V
BALANCED ACCURACY USING DL FOR NOISY TEST DATASET

Models	Binary	Multi
NN	0.811	0.788
CNN	0.844	0.748
ResNet50	0.811	0.762

VI. REFERENCES

REFERENCES

[1] Nandhini Subramanian, “A review of deep learning-based detection methods for COVID-19,” 2022.

[2] Emre AVUÇLU, "COVID-19 detection using X-ray images and statistical measurements," 2022.
[3] Juliana C. Gomes, Valter A. de F. Barbosa, et al., "IKONOS: an intelligent tool to support diagnosis of COVID-19 by texture analysis of X-ray images," 2020.
[4] Geeta Rani, Ankit Misra, et al., "A multi-modal bone suppression, lung segmentation, and classification approach for accurate COVID-19 detection using chest radiographs,"
[5] S. L. Dongguang Li, "An artificial intelligence deep learning platform achieves high diagnostic accuracy for Covid-19 pneumonia by reading chest X-ray images," 2022.
[6] ISM, Chest X-Ray images dataset. [Online]. Available: http://gofile.me/5AIED/xNqNoKPxu
[7] Angel Igareta, Stratified Sampling: You May Have Been Splitting Your Dataset All Wrong. Towards Data Science. [Online]. Available: https://towardsdatascience.com/
[8] Sidheswar Routraya, Arun Kumar Raya, Chandrabhanu Mishrab, "Image denoising by preserving geometric components based on weighted bilateral filter and curvelet transform," 2018.
[9] Mingle Xu, Sook Yoon, et al., "A Comprehensive Survey of Image Augmentation Techniques for Deep Learning," 2023.
[10] NIST/SEMATECH, Handbook of Statistical Methods. NIST/SEMATECH. [Online]. Available: https://www.itl.nist.gov/div898/handbook/index.htm
[11] Haralick texture. [Online]. Available: https://cvexplained.wordpress.com/2020/07/22/10-6-haralick-texture/
[12] Zernike Moments. [Online]. Available: https://cvexplained.wordpress.com/2020/07/21/10-5-zernike-moments/
[13] Pytorch Team, Mobilenet V2. [Online]. Available: https://pytorch.org/hub/pytorch_vision_mobilenet_v2/
[14] Pytorch Team, ResNet. [Online]. Available: https://pytorch.org/hub/pytorch_vision_resnet/
[15] Brett Lantz, Machine Learning with R, 2nd ed. Packt Publishing, 2015.
[16] e. a. Sheng Lu, "Dynamic Weighted Cross Entropy for Semantic Segmentation with Extremely Imbalanced Data," 2019.
[17] Diederik P. Kingma, Jimmy Lei Ba, "Adam: A method for Stochastic Optimization," 2015.