DL GROUP PROJECT - FINAL REPORT
COVID-19 DETECTION BASED ON BREATHING SOUNDS

Vitaliia Marina
vitaliia.marina@tu-ilmenau.de

Ye Yun Khor
ye-yun.khor@tu-ilmenau.de

Ralina Safina
ralina.safina@tu-ilmenau.de

Edwin G. Carreño
edwin-carreno.lozano@tu-ilmenau.de

Project Mentor:
M. Sc. Dominik Walther

January 30, 2022

ABSTRACT

COVID-19 has caused a pandemic of respiratory illness that has negatively affected the economic and social stability of nations worldwide. On the other hand, the impossibility of obtaining updated information on the number of infections and their presence aggravates the situation for effective decision making by governments and health organizations. For those reasons, this work aims to propose a non-invasive strategy that allows a rapid identification of COVID-19 through Convolutional Neural Networks and the use of spectrograms based on audio samples taken from coughing patients (with and without COVID-19). Among implemented architectures the best classification results were performed by Xception model with an accuracy and F1 score of 97.2% and 97% respectively.

1 Introduction

COVID-19 causes distinct pathomorphological alterations in the respiratory system, thus, we are interested to find out whether COVID-19 can be detected based on breathing sounds using neural networks. The input to our algorithm are the spectrogram images of breathing sounds [1]. We plan to use a CNN (Convolutional Neural Network) to output the classification of the breathing sound, i.e., whether it belongs to a COVID-19 or a non COVID-19 patient. All the documentation related to this project including code, images and this report could be found in [2].

2 Related work

Deep learning methods have been used successfully to solve many complex medical problems in the past, so it is not surprising that many researchers have turned to them again in the face of a new challenge. In [3], a group of scientists tackles a problem of detecting COVID-19 from a patient's breathing and coughing patterns through usage of an LSTM(Long Short-Term Memory) model which consists of three LSTM layers. It provided a validation accuracy of 80%. In order to complete their task, they used a dataset from Pfizer Digital Medicine Challenge which consists of raw audio files, and then utilized Mel-frequency cepstral coefficients (MFCCs) to extract relevant features.

The modification of VGG-13 architecture is suggested by the authors in [4]. To prevent overfitting of the model and balance the data two datasets were used: DiCOVA 2021 and augmented positive samples of COUGHVID. By combining cross entropy and focal loss while using an ensemble of two VGG-13 models the validation AUROCs (82.23%) and blind test AUROCs (78.3%) were achieved.

Another interesting approach to tackle the problem of identifying COVID-19 in data samples is related to translating the power-time domain (audio) to frequency-time-power domain (images), in other words using spectrograms as input data

for a CNN architecture. In [5] has been tested the usage of spectrograms in the Xception CNN architecture, achieving a precision of 0.75%-0.80% with a fine-tuning approach as the initial step. However, authors argue that the quality of results is highly related to the quality of the dataset used, high quality images in spectrograms could lead to better results in classification. Other than that, the background noise while recording the breathing sound and the microphone's sensitivity also affect the model [6].

3 Dataset and Features

In this work, a raw dataset [1] consisting of spectrograms of breathing sounds was used. A total of 4424 spectrograms (images) were splitted in two datasets, images associated with identified COVID and Non-COVID cases. Testing dataset contains 87.68% of all samples, whereas the validation dataset contains 12.31% of them. Spectrograms presented in the dataset have different resolutions, they were scaled before feeding them to the neural network. In order to train models and speed up calculations, a preprocessing stage such as resizing (fit images into 200x200, 256x256 and 512x512 sizes) and rescaling was performed.

4 Methods

Figure 1 shows the pipeline used in this project to resolve the task of identifying if an image or a set of them belongs to Non-Covid-19/Covid-19 classes. In general terms, this pipeline consists of collecting data, defining neural network architectures, training models, choosing the model with best performance (without hyperparameter search), tuning the model, reporting results and finally, choosing the model with best metrics. Following subsections describe the key components in the project.

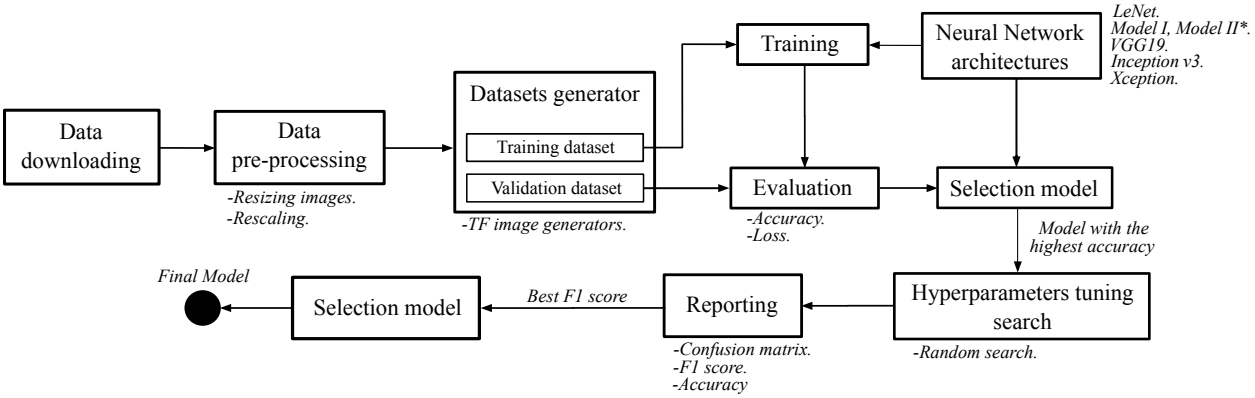


Figure 1: General view of the Pipeline applied for the Covid-19 detection project. *Proposed by authors of this document.

4.1 Neural Networks Architectures

Seven (7) different neural network architectures have been considered to analyze which of them perform in an accurate way the classification task without hyperparameters tuning as a first approximation, and most importantly without incur in a large time consuming training process. Following are the architectures used in this project:

- **LeNet** [7][8].
- **LeNet Modified:** it is essentially the same LeNet architecture but using the ReLU activation function for all the layers previous to the output layer.
- **Model I :** it is a Convolutional Neural Network composed by five (5) layers (input layer, CONV2D of 16 filters with 3x3 kernel size, Dropout layer with a rate of 0.4, GlobalMaxPool2D layer and the output layer with one binary neuron activated with the Sigmoid function).
- **Model II :** it is a Convolutional Neural Network composed by seven (7) layers (input layer, CONV2D of 16 filters with 3x3 kernel size, Max-Pool layer with 2x2 kernel size, CONV2D of 32 filters with 3x3 kernel size, Max-Pool layer with 2x2 kernel size, Dense layer with 64 units and the output layer with one binary neuron activated with the Sigmoid function). Figure 2 shows a more complex architecture compared to the Model I proposed by authors in this document.

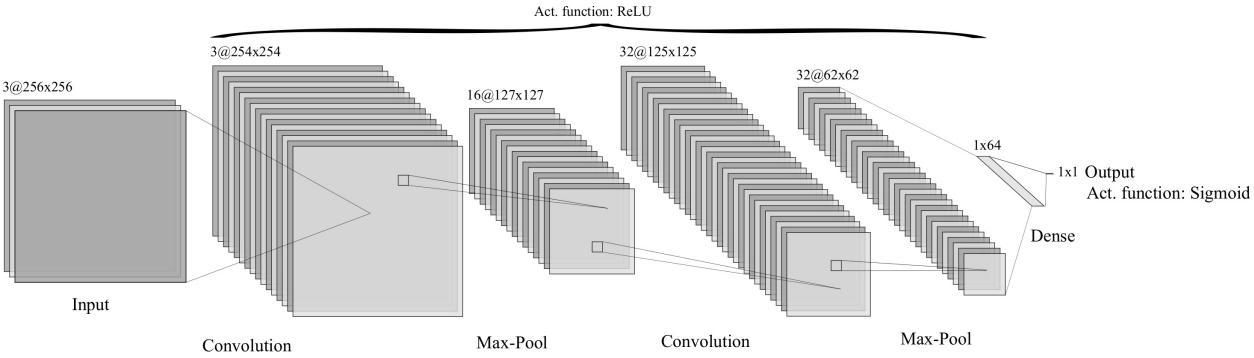


Figure 2: Architecture Model II, proposed by authors in this documents.

- **VGG19 [9]:** it has been implemented using the pre-trained model given in the Keras library [10]. On top of the model, one dense layer (4096 units/layer, ReLu activation) and a binary output (1 unit, Sigmoid activation) were added.
- **Inception v3 [11]:** pre-trained model using in the Keras library. On top of the model, two dense layers (1024 units/layer, ReLu activation) and a binary output(1 unit, Sigmoid activation) were added.
- **Xception [12] :** pre-trained model using in the Keras library. On top of the model, one dense layer (64 units/layer, ReLu activation) and a binary output(1 unit, Sigmoid activation) were added.

4.2 Training

All models have been built and compiled using the Binary Cross Entropy as the loss function, Adam (Adaptive moment estimation) has been chosen as the optimizer by showing great stability in comparison with other optimizers such as SGD(Stochastic Gradient Descent), Adagrad(Adaptive Gradient) and RMSprop(Root Mean Square propagation). A fixed number of epochs has been chosen to analyze the trend in the training stage. Finally, the metric used for evaluating is the accuracy metric. Following are some of the most relevant parameters defined to train the different models before using any tuning technique:

- **Epochs:** fixed to 50 epochs without early stopping.
- **Learning rate:** Default value of 1e-3 before fine tuning.

4.3 Hyperparameters tuning search

This stage consist of using the Keras-Tuner library [13] to find the best combination of hyperparameters that maximizes the accuracy in the chosen model. Finding the sets is an intensive task especially when the space of hyperparamters (all possible combinations of hyperparameters) is large. For that reason, in this work the search has been performed under the following possible values:

- **Number of Units Dense Layers:** 64 or 128.
- **Activation functions:** Tanh, ReLU.
- **Dropout Rate:** 0.2, 0.4, 0.5, 0.8.
- **L2 regularization penalty:** range starting from 0 to 1.0 in steps of 0.2.

5 Experiments/Results/Discussion

Three (3) different kind of stages have been conducted in this project not only to analyze the effect of having different image sizes, but also to choose the best model for the task itself.

5.1 Stage 1. Image Size effect

Image size has a direct impact in the loss and accuracy obtained when models were trained and validated. Three image sizes (200x200, 256x256, 512x512) were chosen to perform experiments, and as a result was obtained that bigger

images tend to improve the accuracy and increase the loss for the models. However, in the Xception model, the loss decreases indicating an additional improvement compared with the other models. Figure 3 shows the loss and accuracy in different architectures mentioned in table 1.

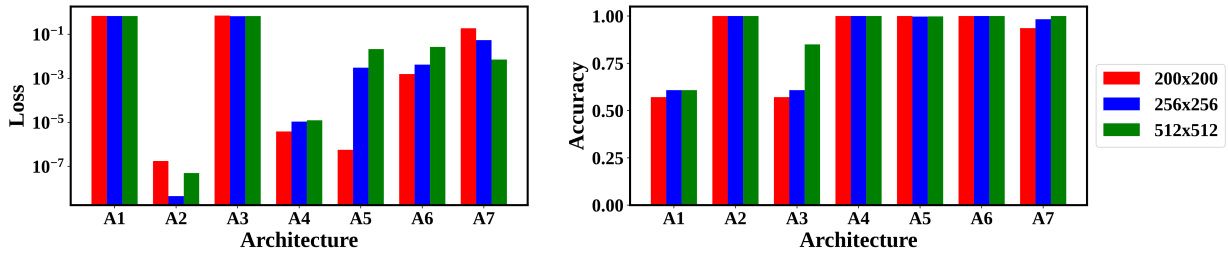


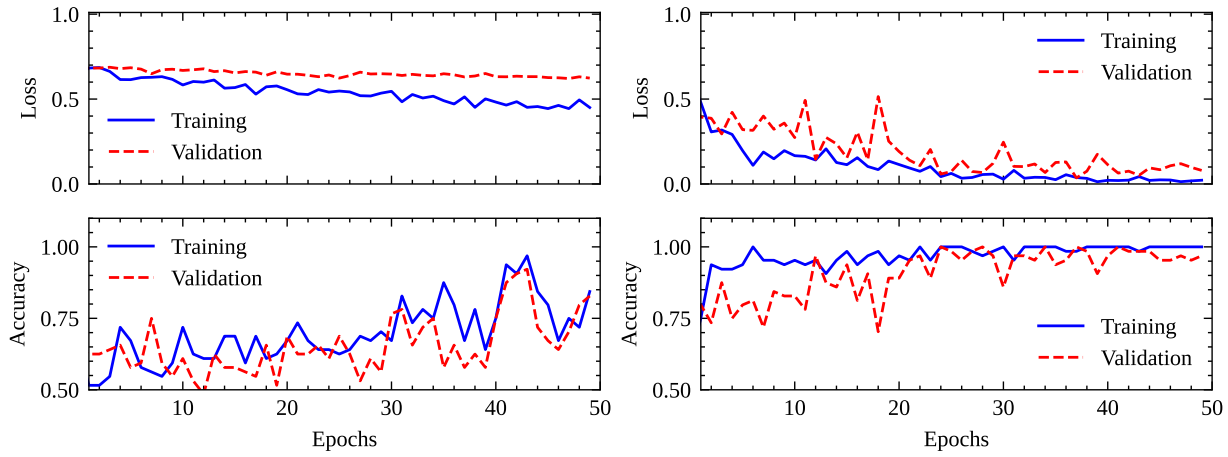
Figure 3: Comparison of loss and accuracy for models trained with images with different sizes.

Table 1:
NOMENCLATURE'S EQUIVALENCE

Nomenclature	A1	A2	A3	A4	A5	A6	A7
Arch. Name	LeNet original	LeNet modified	Model I	Model II	VGG19	Inception v3	Xception

5.2 Stage 2. Finding the Base Architecture

All the models were trained using images with the 256x256 size, in other words, the input shape was defined as 256x256x3 as entry point in each architecture. This image size was chosen to deal with the trade-off existing between training time, accuracy and loss, defining an intermediate point between 200x200 and 512x512. Once the model has been trained, the best model was chosen after evaluating accuracy, and trends in learning curves. Figure 4 shows a comparison between the learning curves in Model I and Xception model; Xception model tends to follow the trend expected in ideal learning curves.



(a) Model I

(b) Xception architecture

Figure 4: Learning curves for Model I (a) and Xception model (b). Notice the first model(a) underfit the data (high bias), whereas the second(b) overfit rapidly but follows an approximate behaviour expected in ideal learning curves.

5.3 Stage 3. Hyperparameters Fine Tuning

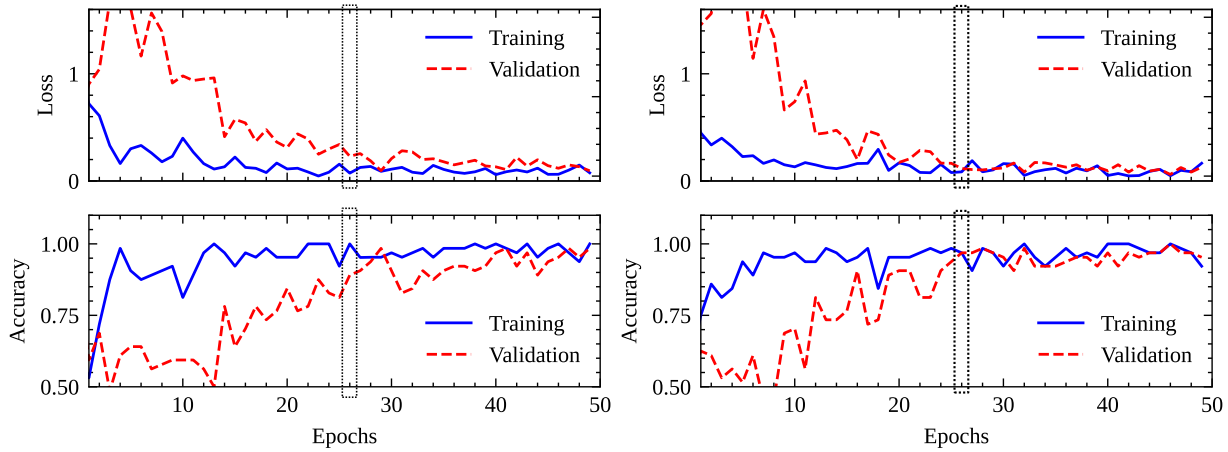
Hyperparameter search has been performed in a sub-space of 100 trials (31.25% of the total number of possible combinations of the hyperparameters mentioned in 4.3). Additionally, the last two layers in the pre-trained model were unlocked to perform a better tuning by taking into account the dataset used in this project. Table 2 summarizes the

loss and accuracy obtained in the hyperparameter search. However, notice that the best experiment(lowest loss and highest accuracy) is not ranked as the first one in the table (see Ranking 1 and 4 in bold font). Following is the set of hyperparameters that maximizes accuracy and produces the lowest error: 64 units, Tanh activation function, Dropout rate of 50%, L2 regularization penalty is 0 and a learning rate starting from 1e-3 to 1.8316e-5 under learning rate scheduler.

Table 2:
RANKING OF THE FIVE BEST MODELS OBTAINED BY KERAS-TUNER

Ranking	Experiment	Loss	Accuracy
-	without (fine tuning)	0.10414	0.97248
1	**Set 1**	**0.13577**	**0.95046**
2	Set 2	0.11396	0.95596
3	Set 3	0.13630	0.93945
4	**Set 4**	**0.09901**	**0.97248**
5	Set 5	0.17731	0.92661

On the other hand, using the set four of hyperparameters (Set 4) not only it is reached a high accuracy (97%) but also decreased the number of epochs required for it by a rate of approx. 50% (compare the accuracy plots in Figure 5 in the number of epochs framed by the dashed rectangles). With the model already tuned, Figure 6 shows a comparison between the ideal confusion matrix for the classifier using the validation dataset (imbalanced). Finally, F1 score could be calculated [14] giving as a result of 97%.



(a) Model with Set 1 (ranked as the best model by Keras-tuner) (b) Model with Set 4 (lowest loss and highest accuracy)

Figure 5: Learning curves for the Xception model with different hyperparameters sets.

		Actual Value	
		Non-Covid	Covid
Predicted Value	Non-Covid	TN 315	FP 0
	Covid	FN 0	TP 230

(a) Ideal confusion matrix.

		Actual Value	
		Non-Covid	Covid
Predicted Value	Non-Covid	TN 310	FP 1
	Covid	FN 14	TP 220

(b) Current confusion matrix after fine tuning.

Figure 6: Confusion matrices for the classifier using Xception architecture on validation dataset(total number of examples is 545).

6 Conclusion/Future Work

Xception model has outperformed the classification task with an accuracy and F1 score of 97.2% and 97% respectively, in comparison with other models reviewed in this document. Complex models such as Xception and Inception v3 extract more properties in images, whereas simpler models tend to extract basics or most notorious features (biasing). Moreover, the image size is a relevant parameter to be considered, down-sampling (resizing with a lower size) decreases accuracy and tends to overfit the data. Higher resolution images tend to occupy more memory and make models to be long time consuming in training, resulting in a trade-off to deal with. Finally, future works could be conducted in studying if the complexity of the Xception model could be reduced using an hybrid implementation(time series) that takes advantage of the time property of spectrograms.

7 Contributions

- **Vitaliia Marina:** Methods, pipeline, Model I architecture.
- **Edwin Carreño** - Methods, pipeline, design of experiments, hyperparameter tuning, results & reporting.
- **Ralina Safina** - Data preprocessing, Model II architecture, results & reporting.
- **Ye Yun Khor** - Hyperparameter tuning, results & reporting and video script.

References

[1] Unais Sait, Gokul Kv, Shivakumar Sanjana, Tarun Kumar, Rahul Bhaumik, Sunny Prakash Prajapati, and Kriti Bhalla. Spectrogram images of breathing sounds for covid-19 and other pulmonary abnormalities. `https://data.mendeley.com/datasets/pr7bgzxpgv/1`, 2021. https://doi.org/10.17632/pr7bgzxpgv.2.

[2] Viitalia Marina, Edwin Carreno, Ralina Safina, and Y. Yun Khor. Covid19 detection using spectrograms. `https://github.com/imciflam/covid-detection`, 2021.

[3] Lazhar Khriji, Ahmed Ammari, Seifeddine Messaoud, Soulef Bouaafia, Amna Maraoui, and Mohsen Machhout. Covid-19 recognition based on patient's coughing and breathing patterns analysis: Deep learning approach. In *2021 29th Conference of Open Innovations Association (FRUCT)*, pages 185–191, 2021.

[4] Sunil Rao, Vivek Narayanaswamy, Michael Esposito, Jayaraman Thiagarajan, and Andreas Spanias. Deep learning with hyper-parameter tuning for covid-19 cough detection. In *2021 12th International Conference on Information, Intelligence, Systems Applications (IISA)*, pages 1–5, 2021.

[5] Ciro Rodriguez Rodriguez, Daniel Angeles, Renzo Chafloque, Freddy Kaseng, and Bishwajeet Pandey. Deep learning audio spectrograms processing to the early covid-19 detection. In *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 429–434, 2020.

[6] Unais Sait, Gokul Lal K.V., Sanjana Shivakumar, Tarun Kumar, Rahul Bhaumik, Sunny Prajapati, Kriti Bhalla, and Anaghaa Chakrapani. A deep-learning based multimodal system for covid-19 diagnosis using breathing sounds and chest x-ray images. *Applied Soft Computing*, 109:107522, 2021.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Aston Zhang, Zack C. Lipton, Mu Li, Alexa J. Smola, Brent Wernness, Rachel Hu, Shua Zhang, Yi Tay, Anirudh Dagar, and Yuan Tang. *Dive into Deep Learning*. D2L.ai, 2022. `https://d2l.ai/chapter_convolutional-neural-networks/lenet.html`.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[10] Francois Chollet et al. Keras: Deep learning for humans. `https://github.com/keras-team/keras`, 2022.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[12] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[13] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. `https://github.com/keras-team/keras-tuner`, 2019.

[14] A. Burkov. *The Hundred-Page Machine Learning Book*, chapter 5, pages 13–19. Andriy Burkov, 2019. `http://themlbook.com/wiki/doku.php`.