

People's Democratic Republic of Algeria  
Ministry of Higher Education and Scientific Research  
University Of Ferhat Abbas Setif -1-  
Departement Of Computer Science



---

---

## Machine Learning based Models for Audio Signal Classification: Application to COVID-19 Detection

---

---

Ph.D. Thesis by: SKANDER HAMDI

Department of Computer science

UNIVERSITY OF FERHAT ABBES SETIF -1-

This thesis satisfies the requirements for the degree of Ph.D. in  
Computer Science - Machine Learning and Intelligent Systems

---

---

### Committee of examiners

MOUSSAOUI Abdelouahab	Prof.	Univ. Ferhat Abbes Setif 1	President
SAIDI Mohamed	MCA.	Univ. Ferhat Abbes Setif 1	Supervisor
OUSSALAH Mourad	HDR.	Univ. Oulu, Finland	Co-Supervisor
ATTIA Abdelouahab	MCA.	Univ. BBA	Examiner
TOUMI Lyazid	MCA	Univ. Ferhat Abbes Setif 1	Examiner

PUBLICLY DEFENDED ON: NOVEMBER, 28 2023



## تلخيص

تقنية تعلم الآلة قد غيّرت مجال تشخيص الأمراض الطبية، وأصبحت أهميتها أكثر وضوحًا، حيث يمكنها تحليل مختلف البيانات الطبية من مصادر متنوعة مثل الصور الطبية، ونتائج الاختبارات المخبرية، والسجلات السريرية، والإشارات الفيزيولوجية. منذ ظهور تفشي فيروس كوفيد-19، تم استخدام بعض التقنيات مثل اختبار تفاعل البوليمراز المتسلسل للنسخ العكسي واختبار الأجسام المضادة السريعة للتشخيص وذلك لعزل المرضى بسرعة لمنع انتشار الفيروس. ومع ذلك، يتسبب نقص أدوات الاختبار والوقت اللازم للحصول على نتائج الاختبار الذي يكون في بعض الأحيان طويلًا، في وضع حرج يستدعي بناء أدوات تشخيص مبنية على الذكاء الاصطناعي لتحديد هوية المصابين بسرعة. تم إجراء العديد من الدراسات في هذا الصدد، لبناء أنظمة تشخيص قائمة على تعلم الآلة، حيث تم استخدام المعلومات السريرية والسوابق الطبية لتوقع وجود المرض، ومع ذلك، فإن هذه المعلومات قد لا تكون كافية لتشخيص دقيق. تمت دراسة تقنيات أخرى مثل صورة أشعة الصدر وفحوصات الأشعة المقطعية وأظهرت أداءً عاليًا وواعدًا، ومع ذلك، فإن هذه الأساليب تتطلب اتصالًا مباشرًا بين العاملين في مجال الرعاية الصحية والمرضى ولا تكون دائمًا عملية وآمنة، خاصة خلال جائحة كوفيد-19 المعدية بشكل كبير. وبالتالي، هناك حاجة ملحّة لأنظمة تشخيص عن بُعد قائمة على تعلم الآلة يمكنها التعرف على المصابين بسرعة وبدقة دون الحاجة إلى اتصال مباشر. يمكن أن يظهر فيروس كوفيد-19 مع مجموعة من الأعراض مثل الحمى والتعب والسعال، ويُعتبر السعال أحد الأعراض الشائعة جدًا. استخدام بيانات السعال أو الصوت التنفسي في عملية التشخيص قد أظهر أداءً ونتائج تشخيص سريعة وعالية الدقة. وبالتالي، فقد أكد ظهور فيروس كوفيد-19 أهمية تطوير وتعزيز أنظمة تشخيص قائمة على تعلم الآلة يمكنها استخدام بيانات السعال، أو الصوت التنفسي بصفة عامة لتوفير تشخيص فعال، سريع وعن بُعد للمرض.

تم في هذه الأطروحة اقتراح عدة طرق جديدة لفحص فيروس كوفيد-19، حيث يتم استخدام صوت السعال لتدريب نماذج مختلفة. تعتمد الطريقة الأولى على دمج شبكة عصبية إلتفافية مع أخرى ذات الذاكرة الطويلة قصيرة المدى مع آلية التركيز بالإضافة إلى تعزيز البيانات الطيفية. أما الطريقة الثانية فهي عبارة عن مجموعة من تقنيات تعلم التجميع، حيث يتم استخدام خوارزمية الغابات العشوائية بشكل أساسي. تتكون الطريقة الأولى من إرسال مجموعة السمات ذات المستوى المنخفض بعبء الأصلي إلى المصنف، بينما تعتمد الطريقتين الثانية والثالثة على ضغط مساحة السمات وتقليل الأبعاد باستخدام التشفير التلقائي المكسب وتضمين خطي محلي، على الترتيب، لتقليل تعقيد الحساب واستخدام السمات التمييزية الأكثر قوة تفرقة لأداء مهمة التصنيف. يتم تقييم الطرق المقترحة على مجموعة بيانات متاحة للعامة تُسمى COUGHVID، وتظهر النتائج فعالية الطرق المقترحة وأدائها العالي. تتضمن مساهمات هذه الأطروحة تطوير تقنيات جديدة مبنية على تعلم الآلة العميق والتعلم الآلي للتشخيص لفحص فيروس كوفيد-19، والتي يمكن استخدامها في الإعدادات السريرية وعن بُعد على حد سواء. تسلط هذه النتائج الضوء على إمكانية تشخيص فيروس كوفيد-19 بسرعة ودقة باستخدام تقنيات تعلم الآلة وقدرتها على مساعدة مقدمي الرعاية الصحية في اتخاذ القرارات، خاصةً خلال فترة الجائحة.



## RÉSUMÉ

L'apprentissage automatique a transformé le domaine du diagnostic médical, et son importance est devenue encore plus évidente, où il peut analyser différentes modalités de données médicales de sources diverses, telles que les images médicales, les résultats des tests de laboratoire, historiques cliniques et les signaux physiologiques. Depuis l'apparition de l'épidémie de COVID-19, des techniques telles que le RT-PCR et les tests rapides d'antigènes ont été utilisées pour le diagnostic afin d'isoler rapidement les patients et de prévenir la propagation de l'infection. Cependant, le manque des kit de dépistage et la durée de ces techniques présentent une état d'urgence qui nécessite la création d'outils de diagnostic basés sur l'IA pour identifier et isoler les patients rapidement. De nombreuses études ont été menées à cet égard pour développer des systèmes de diagnostic basés sur l'apprentissage automatique, où les informations et les historiques cliniques ont été utilisés pour prédire la présence de la maladie, cependant, ces informations peuvent ne pas être suffisantes pour un diagnostic précis. D'autres techniques telles que la radiographie pulmonaire et les scanners CT ont été étudiées et ont montré des performances élevées et prometteuses. Cependant, ces méthodes nécessitent un contact physique entre les professionnels de la santé et les patients, qui peut présente une éventuelle propagation du virus. Par conséquent, il est essentiel de disposer de systèmes de diagnostic à distance qui peuvent identifier rapidement, en toute sécurité et avec précision les personnes infectées sans nécessiter de contact physique. COVID-19 peut se manifester par un ensemble de symptômes tels que la fièvre, la fatigue et la toux, qui est considérée comme l'un des symptômes les plus courants. L'utilisation des données de toux ou de sons respiratoires dans le processus de diagnostic a montré des performances acceptables et des résultats rapides. Donc, l'apparition de la COVID-19 a signalé l'importance du développement et de l'amélioration des systèmes de diagnostic qui peuvent utiliser les données de toux ou de sons respiratoires pour fournir un diagnostic efficace, rapide et à distance.

Dans cette thèse, nous proposons des méthodes novatrices pour le dépistage de la COVID-19, où le son de la toux est utilisé avec des techniques différentes. Notre première approche est basée sur l'hybridation d'un réseau de neurones convolutifs (CNN) avec une mémoire à court et long terme (LSTM) ainsi qu'un mécanisme d'attention et une augmentation des données spectrales. La deuxième approche est un ensemble de méthodes basées sur l'apprentissage ensembliste, où l'algorithme des Forêts aléatoires est principalement utilisé. La première approche consiste à transmettre le vecteur de descripteurs de bas niveau initial au classifieur, la deuxième et la troisième sont basées sur la compression de l'espace des attributs et la réduction de la dimensionnalité en utilisant des autoencodeurs et l'encastrement linéaire local, respectivement, pour réduire la complexité de calcul et utiliser les attributs les plus discriminantes. Les méthodes proposées sont évaluées sur un ensemble de données disponible publiquement appelé COUGHVID, et les résultats démontrent leur faisabilité et leur haute performance. Les contributions de cette thèse comprennent le développement de nouvelles architectures basées sur l'apprentissage profond et l'apprentissage ensembliste pour le dépistage de la COVID-19, qui peuvent être utilisées à la fois dans des contextes cliniques et à distance. Ces résultats mettent en évidence le potentiel du diagnostic basé sur l'apprentissage automatique pour améliorer la rapidité et la précision du dépistage du COVID-19 et son importance pour la prise des décisions éclairées, notamment pendant les pandémies.



## ABSTRACT

Machine learning (ML) has transformed the field of medical diagnosis, and it has become even more important, where it can analyze different medical-related data modalities from various sources, such as medical images, laboratory test results, clinical histories, and physiological signals. Since the emergence of COVID-19 outbreak, techniques such as RT-PCR and rapid antigen test have been used for diagnosis in order to quickly isolate patients to prevent virus spread, however, the lack of test kits and time-consuming of these techniques cause a serious situation that requires building some AI-based diagnosis tools to quickly identify and isolate infected people. Many studies have been conducted in this respect, to build ML-based diagnosis systems, where clinical information and history have been used to predict the presence of the disease, however, this information may not be sufficient for accurate diagnosis. Other techniques like Chest X-ray and CT scans have been investigated and showed a high and promising performance, however, the mentioned methods require physical contact between healthcare workers and patients and they are not always practical and safe, especially during the highly contagious COVID-19 pandemic. Therefore, there is a critical need for remote, ML-based diagnosis systems that can quickly, safely, and accurately identify infected individuals, without requiring physical contact. COVID-19 can present with a range of symptoms, such as fever, fatigue, and, coughing, which is considered one of the most common symptoms. Utilizing cough or respiratory sound data in the diagnosis process has shown promising performance and quick diagnosis results. Thus, the emergence of COVID-19 has underscored the importance of developing and enhancing ML-based diagnosis systems that can utilize cough or respiratory sound data to provide an efficient, quick, and remote diagnosis of the disease.

In this thesis, we propose novel methods for COVID-19 screening, where cough sound is used to employ various techniques. Our first approach is based on hybridizing Convolutional Neural Network (CNN) with Long-Short Term Memory (LSTM) along with Attention mechanism and spectral data augmentation. The second one is a set of ML approaches that are based on ensemble learning, where Random Forest algorithm was primarily used. The first approach consists of forwarding a raw Low-Level Descriptors (LLD) vector to the classifier, the second and the third are based on feature space compression and dimensionality reduction using Stacked Autoencoders and Locally Linear Embedding (LLE), respectively, to reduce computing complexity and make use of the most discriminant features to perform the classification task. The proposed methods are evaluated on a publicly available dataset called COUGHVID, and the results demonstrate their feasibility and high performance. The contributions of this thesis include the development of novel Deep Learning (DL) and ML-based architectures for COVID-19 screening, which can be used in both clinical and remote settings. These findings highlight the potential of ML and DL-based diagnosis system in improving the speed and accuracy of COVID-19 screening and its potential to assist healthcare providers for decision making, especially during pandemics.





## DEDICATION AND ACKNOWLEDGEMENTS

First of all, Thank Allah, who has given me the ability, motivation, patience, and resources to complete this journey. I am grateful for his guidance and support throughout this process.

الحمد لله الذي بنعمته تم الصالحات

I would like to express my deepest gratitude and appreciation to my supervisors. Their guidance, support, help, and encouragement have been invaluable to me throughout my Ph.D. program.

I would also like to pay tribute to my late father, may Allah grant him his paradises. He was the most important person who encouraged me to move on and complete the Ph.D. degree, and his memory will always be cherished. I hope he is proud of me.

I am indebted to my mother for her unwavering love, encouragement, and support throughout my academic journey. Her prayers and sacrifices have been instrumental in helping me achieve this milestone.

I would like to thank my brothers, Naoufel and Rostom, for their unwavering support and encouragement throughout my Ph.D. program.

My heartfelt appreciation goes to my wife Mafaza, who has been a constant source of love, encouragement, and support. Her unwavering belief in me has been a great source of strength and motivation.

I would also like to express my gratitude to my best friends, Mohamed and Ayoub, for their continuous support and encouragement. I wish them all the best in their achievements and future endeavors.

I am grateful to all my teachers from the beginning of my journey at the university to this day. Their knowledge, expertise, and guidance have been much important to my academic development.

Finally, I extend a special thanks to the President of the committee and examiners for accepting the examination of my modest work. Their constructive criticism and feedback will have a significant impact on improving the quality of this thesis.

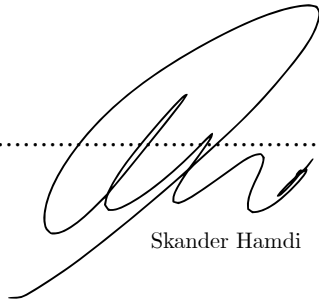


## AUTHOR'S DECLARATION

I, Skander HAMDI, hereby declare that the thesis entitled "Machine Learning based Models for Audio Signal Classification: Application to COVID-19 Detection" and the work presented in it are my original research, and I confirm the following:

- This work was conducted while I was a candidate for a research degree at the University of Ferhat Abbas Setif -1- since December 2019.
- There is no part of this thesis has been previously submitted for a another degree at this University or any other institution.
- Every other's consulted work has been cited and attributed by reference.
- The content of this thesis is solely the result of my original research and analysis.
- I have acknowledged all sources of assistance.
- I acknowledge that I have made use of some AI and grammar tools to make certain corrections and improve this manuscript.

SIGNED: ..... DATE: 12/12/2023 .....



Skander Hamdi



## TABLE OF CONTENTS

	<b>Page</b>
<b>List of Tables</b>	<b>xv</b>
<b>List of Figures</b>	<b>xvii</b>
<b>Acronyms</b>	<b>xxi</b>
<b>List of Algorithms</b>	<b>xxv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 COVID-19 outbreak . . . . .	1
1.2 Testing and diagnosing tools . . . . .	2
1.3 Machine Learning (ML) in COVID-19 Diagnosis . . . . .	3
1.4 COVID-19 Symptoms . . . . .	4
1.5 Motivations . . . . .	4
1.6 Thesis objectives and research questions . . . . .	6
1.7 Thesis outline . . . . .	7
1.8 Publications . . . . .	8
<b>2 Theoretical background</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 Introduction to Machine Learning . . . . .	11
2.2.1 Types of Machine Learning . . . . .	12
2.2.2 Machine learning pipeline . . . . .	13
2.3 Common machine learning algorithms . . . . .	15
2.3.1 Linear and polynomial regression . . . . .	16
2.3.2 Support Vector Machine (SVM) . . . . .	17
2.3.3 Decision Trees . . . . .	18
2.3.4 K-means . . . . .	19
2.4 Ensemble Learning . . . . .	20
2.4.1 Bagging . . . . .	20
2.4.2 Boosting . . . . .	20

## TABLE OF CONTENTS

---

2.4.3	Stacking . . . . .	21
2.4.4	Random Forest . . . . .	21
2.5	Introduction to Deep Learning . . . . .	23
2.5.1	Artificial Neural Network (ANN) . . . . .	23
2.5.2	Backpropagation algorithm . . . . .	26
2.5.3	Convolutional Neural Network (CNN) . . . . .	26
2.5.4	Long-Short Term Memory (LSTM) . . . . .	31
2.5.5	Attention Mechanism . . . . .	33
2.6	Overfitting and Normalization . . . . .	34
2.6.1	Dropout . . . . .	34
2.6.2	Batch Normalization (BN) . . . . .	35
2.7	Overview on Audio Features . . . . .	35
2.7.1	Time domain features . . . . .	36
2.7.2	Frequency domain features . . . . .	37
2.8	Dimensionality Reduction . . . . .	41
2.8.1	Locally Linear Embedding (LLE) . . . . .	41
2.8.2	Autoencoders . . . . .	43
2.9	Data Augmentation Methods . . . . .	44
2.9.1	Synthetic Minority Oversampling TEchnique (SMOTE) . . . . .	45
2.9.2	Audio data Augmentation . . . . .	45
2.10	ML and DL for respiratory diseases diagnosis . . . . .	49
2.11	Conclusion . . . . .	52
<b>3</b>	<b>Literature Review</b> . . . . .	<b>53</b>
3.1	Introduction . . . . .	53
3.2	Cough and respiratory sounds under COVID-19 data availability . . . . .	54
3.2.1	COUGHVID dataset . . . . .	54
3.2.2	University of Cambridge (UC) dataset . . . . .	54
3.2.3	Coswara dataset . . . . .	55
3.3	Related works . . . . .	56
3.3.1	ML-based Approaches . . . . .	57
3.3.2	Deep Learning (DL)-based Approaches . . . . .	62
3.3.3	Works involved comparison: ML and DL Techniques . . . . .	71
3.4	Comparative analysis . . . . .	75
3.4.1	ML vs. DL . . . . .	75
3.4.2	Feature extraction and selection methods . . . . .	76
3.4.3	Data augmentation strategies . . . . .	77
3.5	Identified limitations . . . . .	78
3.6	Statistical summary of the literature review . . . . .	79

3.7	Conclusion and proposed approaches . . . . .	81
<b>4</b>	<b>A-CNN-LSTM and SpecAugment framework</b>	<b>83</b>
4.1	Introduction . . . . .	83
4.2	Materials and methods . . . . .	84
4.2.1	Dataset description . . . . .	84
4.2.2	Data cleaning and preparation . . . . .	85
4.2.3	Mel-Spectrogram computation . . . . .	88
4.2.4	Input size standardization . . . . .	91
4.2.5	Two-level data augmentation . . . . .	91
4.2.6	Attention-based Hybrid CNN-LSTM Architecture . . . . .	93
4.2.7	Baselines . . . . .	97
4.3	Experimental Design . . . . .	97
4.3.1	Hyper-parameters optimization . . . . .	98
4.3.2	Models training . . . . .	98
4.3.3	Performance evaluation . . . . .	99
4.4	Results and findings . . . . .	99
4.4.1	Baselines . . . . .	99
4.4.2	Proposed approach . . . . .	100
4.5	Analysis and interpretation . . . . .	100
4.6	Limitations of the approach . . . . .	105
4.7	Concluding notes . . . . .	106
<b>5</b>	<b>ML approaches based on ensemble learning and LLD</b>	<b>109</b>
5.1	Introduction . . . . .	109
5.2	Data exploratory and feature engineering . . . . .	110
5.2.1	Data preparation . . . . .	110
5.2.2	Low-Level Descriptors (LLD) ComParE 2016 feature set extraction . . . . .	110
5.2.3	Exploratory Data Analysis . . . . .	111
5.2.4	Class balancing by oversampling: SMOTE . . . . .	113
5.3	Approach evaluation and validation . . . . .	115
5.4	Full-length LLD feature vector . . . . .	116
5.4.1	Methodology description and model training . . . . .	116
5.4.2	Results and discussion . . . . .	117
5.5	Bottleneck-based feature extraction . . . . .	118
5.5.1	Methodology description and model training . . . . .	119
5.5.2	Results and discussion . . . . .	119
5.6	LLE dimensionality reduction . . . . .	122
5.6.1	Methodology description and model training . . . . .	122

## TABLE OF CONTENTS

---

5.6.2	Results and discussion . . . . .	123
5.7	Analysis and comparison . . . . .	125
5.8	Ablation study . . . . .	128
5.8.1	Effectiveness of Random Forest and Ensemble classifiers . . . . .	129
5.8.2	Effectiveness of LLE . . . . .	130
5.8.3	Reduced feature analysis . . . . .	131
5.9	Concluding notes . . . . .	132
<b>6</b>	<b>General conclusion</b>	<b>135</b>
	<b>Bibliography</b>	<b>141</b>
	<b>Appendix: Diagnosis mobile application</b>	<b>155</b>
	Introduction . . . . .	155
	Technical Implementation Details . . . . .	155
	Systematic Overview of the Developed System . . . . .	155
	Diagnosis Flow and Use Cases . . . . .	156



## LIST OF TABLES

TABLE	Page
3.1 COVID-19 status label distribution over the three statuses <i>Healthy</i> , <i>COVID-19</i> , and <i>Symptomatic</i> , which are reported by users . . . . .	55
3.2 Coswara status label distribution over COVID-19 statuses <i>Healthy</i> , and <i>Not-healthy</i> , by excluding the noisy and highly degraded audio recordings . . . . .	55
3.3 Summary of ML-based studies. DA refers to Data Augmentation, CB refers to Class Balancing, dash (-) refers to not reported (used and not mentioned, or not completely used), <i>Acc</i> for accuracy, <i>Rec</i> for recall, <i>AUC</i> for Receiver Operating Characteristic Area Under Curve (ROC-AUC), <i>Prec</i> for precision, <i>Spec</i> for specificity . . . . .	61
3.4 Summary of DL-based studies. . . . .	68
3.5 Summary of Comparative Studies between ML and DL Techniques . . . . .	74
3.6 Comparative results with and without data augmentation or class balancing . . . . .	79
4.1 COVID-19 status label distribution over the three statuses <i>Healthy</i> , <i>COVID-19</i> , and <i>Symptomatic</i> after removing <i>No-COVID-19 status</i> and <i>empty signals</i> during silence removal step . . . . .	86
4.2 COVID-19 status label distribution over the three statuses <i>Healthy</i> , <i>COVID-19</i> , and <i>Symptomatic</i> after omitting the recordings with <i>cough_detected</i> value under 0.7 . . . . .	87
4.3 The new classes namely <i>Likely-COVID-19</i> and <i>Non-Likely-COVID-19</i> after combining <i>Positive COVID-19</i> and <i>Symptomatic</i> classes . . . . .	88
4.4 The proposed network architecture in terms of layer, output size. Each block (CNN, LSTM, Attention and Full connection) is surrounded by horizontal lines. . . . .	96
4.5 Mel-spectrogram computation, data augmentation and classification model architecture hyper-parameters tuning . . . . .	98
4.6 Results of 10-fold cross-validation for the proposed baselines (LSTM only, CNN only, and hybrid CNN-LSTM) using the aforementioned evaluation metrics. . . . .	100
4.7 The result of each fold of the 10-fold Cross-validation (Cross-Validation) for the novel approach (Attention-based Hybrid CNN-LSTM) in term of the above-mentioned evaluation metrics. . . . .	102

LIST OF TABLES

---

4.8 The performance in terms of the six evaluation metrics of our novel approach on the unseen data compared to the proposed baselines *A-CNN-LSTM* refers to Attention-based Hybrid CNN-LSTM . . . . . 103

4.9 The performance of our model with the proposed data augmentation (Pitch shifting (P-S) and SpecAugment) against pitch shifting only . . . . . 104

5.1 Overall statistics about COUGHVID dataset at this stage (No-status elimination, Silence removal, and Multi-class to binary classification problem). Total length in *hours*, where maximum, minimum, and mean in *seconds*. . . . . 110

5.2 ComParE feature set [106, 107, 142] . . . . . 111

5.3 Tuned hyper-paramaters for the first approach LLD-RF.  $\sqrt{F'}$  presents the square root of  $F$  which is the total number of features (6373) . . . . . 116

5.4 5-fold Cross-Validation and testing results of the first approach LLD-RF . . . . . 117

5.5 Tuned hyper-parameters for the second approach Bottleneck-LLD-RF where  $\sqrt{F'}$  presents the square root of the latent space (1024) . . . . . 120

5.6 5-fold Cross-Validation and testing results of the second approach Bottleneck-LLD-RF 121

5.7 Selected hyper-parameters for the third approach LLE-LLD-RF where  $\sqrt{F'}$  presents the square root of the LLE reduced vector (3187) . . . . . 123

5.8 5-fold Cross-Validation and testing results of the third approach LLE-LLD-RF . . . . 124

5.9 Comparison of 5-fold CV and testing results of the three approaches, the first block of the table presents the Cross-Validation results, shown as *avg ± sd*, while the second block shows testing results on the same testing set. The best testing results are highlighted with bold text . . . . . 125

5.10 The time cost of the three approaches for training (explained in seconds) and inference (explained in millisecond) . . . . . 128

5.11 First part of the ablation study which involves training different classifiers to demonstrate the performance of the presented study. . . . . 129

5.12 Second part of the ablation study which involves training our Random Forest classifier using the embeddings and components generated by 6 dimensionality reduction techniques. . . . . 130

## LIST OF FIGURES

FIGURE	Page
2.1 Overview of the ML Pipeline, showing the various stages, from data collection to model prediction . . . . .	16
2.2 General structure of a DNN with input, output, and three hidden layers. Each unit in each layer processes input using an activation function and forwards its activation to the next layer. . . . .	24
2.3 A schematic diagram of a convolutional layer that shows the receptive field of a neuron and how weights are shared within the receptive field. . . . .	28
2.4 Convolution operation between input (local receptive field) and filter to produce feature maps. . . . .	29
2.5 Example of pooling operations in CNNs. Max pooling selects the maximum value in each pooling window, while min pooling selects the minimum value and average pooling calculates the average value. . . . .	30
2.6 General flow of a CNN architecture . . . . .	30
2.7 General structure of LSTM cell . . . . .	33
2.8 Illustration of a sound wave showing the variation of amplitude over time. The zoomed-in section highlights the detailed variations in signal amplitude. . . . .	36
2.9 Illustration of a sound wave and the spectrum which is the result of applying Fast Fourier Transform (FFT). . . . .	38
2.10 Illustration of the Short-Time Fourier Transform (STFT) applied on an input signal. The input signal is divided into overlapping windows of fixed length (window size) and shifted by a fixed interval (hop length) to produce a sequence of windowed segments. Each segment is windowed using a window function and then, spectrum is computed using FFT with a chosen FFT length. The amount of overlap between adjacent windows (overlap length) affects the time-frequency resolution trade-off in the resulting <i>spectrogram</i> . . . . .	39
2.11 Example of an input signal wave and the resulting <i>Spectrogram</i> , where the y-axis is converted to log-scale and the color dimension to decibels. . . . .	40
2.12 Example of an input spectrogram and the resulting <i>Mel-spectrogram</i> . . . . .	40
2.13 Illustration of the three main steps of LLE [101]. . . . .	43

LIST OF FIGURES

---

2.14 General structure of a deep autoencoder with two hidden layers, a latent space of size 3, and an input size of 6. The encoder maps the input to a compressed representation in the latent space, while the decoder maps the latent space back to the original input space. . . . . 44

2.15 Pitch-shifting transformation with  $\alpha = 3$  . . . . . 46

2.16 Time-shifting transformation with  $n_0 = 0.5$  . . . . . 46

2.17 Time-stretching transformation with  $s = 0.5$  . . . . . 47

2.18 White noise addition with a standard deviation  $\sigma = 0.02$ . . . . . 47

2.19 Simple illustration of SpecAugment technique. . . . . 49

3.1 Histogram for the class label distribution of the three datasets *COUGHVID*, *UC*, and *Coswara* . . . . . 56

3.2 Number of published papers, related to COVID-19 diagnosis from respiratory sound and more specifically, cough, per year starting from 2020 until 2023, including the number of excluded which were excluded from the review. . . . . 80

3.3 (a) Number of published papers per category, ML-based, DL-based, and comparative studies between ML and DL; (b) Number of papers per data augmentation and class balancing type, including synthetic-based methods, signal-based augmentation, spectral-based augmentation, and others. . . . . 80

4.1 Overview of the proposed framework. . . . . 84

4.2 Example of the applied silence removal from an audio signal . . . . . 86

4.3 Audio waveform and corresponding mel-spectrogram for two *Non-Likely-COVID-19* cases (1 and 2) and two *Likely-COVID-19* cases (1 and 2), with each region of interest labeled H11, H12, H13, H14, H15, and H16 for *Non-Likely-COVID-19* case (1); H21, H22, and H23 for *Non-Likely-COVID-19* case (2); C11, C12, and C13 for *Likely-COVID-19* case (1); and C21 and C22 for *Likely-COVID-19* case (2). The x-axis represents time in seconds, while the y-axis represents frequency in Hz. The color scale indicates the magnitude in dB . . . . . 90

4.4 An example of two audio signals, the first with length 58275 (2.64s) which has been zero-padded to 7.07s, and the second with a length of 200407 (9.08s) which has been truncated to 7.07s . . . . . 91

4.5 Utilizing pitch shifting technique with  $n\_step = -4$  on a *Likely-COVID-19* audio signal with the corresponding mel-spectrogram (original and augmented) with a highlight of the high magnitude regions. . . . . 92

4.6 Illustration of SpecAugment for a Non-Likely-COVID-19 sample, a new mel-spectrogram is generated by combining *Frequency-masking* and *Time-masking*, for each mel-spectrogram in the dataset. . . . . 94

4.7	Illustration of SpecAugment for a Likely-COVID-19 sample, two new mel-spectrograms are generated by combining <i>Frequency-masking</i> and <i>Time-masking</i> , for each mel-spectrogram in the dataset. . . . .	94
4.8	Block diagram of our proposed Attention-based hybrid CNN-LSTM architecture . . .	95
4.9	Accuracy and loss curves for baseline, (a) and (b) for accuracy and loss curves of CNN model, (c) and (d) for LSTM, where (e) and (f) for hybrid CNN-LSTM . . . . .	101
4.10	ROC curves of the three proposed baselines CNN, LSTM, and Hybrid CNN-LSTM which were computed from the model's output probabilities . . . . .	102
4.11	(a) Accuracy, (b) loss, (c) sensitivity, and (d) precision curves for our novel proposed approach Attention-based hybrid CNN-LSTM . . . . .	103
4.12	ROC curves of the baselines and our proposed approach. (a) displays the full ROC curves, while (b) zooms in on the top-left corner to highlight the differences. . . . .	104
4.13	Confusion matrices for the test set: (a) LSTM model and (b) CNN model . . . . .	104
4.14	Confusion matrices for the test set: (a) Hybrid CNN-LSTM model and (b) Proposed Attention-based Hybrid CNN-LSTM model . . . . .	105
5.1	Box and whisker plot of the six most important features extracted using Recursive Feature Elimination (RFE). No significant differences between the two classes are observed, and multiple outliers are present. This suggests the potential need to further data engineering investigation in the dataset to find a discrimination between the two classes. . . . .	112
5.2	Scatterplot matrix of the first three embeddings of LLE where diagonal shows the distribution of embeddings values using Kernel Density Estimation (KDE), illustrating a trend of high density for Non-Likely-COVID-19 class . . . . .	113
5.3	Scatterplots of PC1 and PC2 before (a) and after (b) SMOTE to show similar data distribution after oversampling the minority class . . . . .	114
5.4	Scree plot showing the explained variance ratio of the top 10 principal components before and after oversampling the minority class using SMOTE . . . . .	115
5.5	Systematic overview of the LLD-RF approach . . . . .	116
5.6	Training and Cross-Validation learning curves of first approach LLD-RF . . . . .	117
5.7	LLD-RF performance evaluation, (a) shows the ROC curve of the predicted probabilities and (b) presents the confusion matrix of the test set . . . . .	118
5.8	Block diagram of the second approach Bottleneck-LLD-RF . . . . .	119
5.9	Autoencoder reconstruction loss curves for training and validation . . . . .	120
5.10	Training and Cross-Validation learning curves of second approach Bottleneck-LLD-RF	121
5.11	Bottleneck-LLD-RF performance evaluation, (a) shows the ROC curve of the predicted probabilities, and (b) presents the confusion matrix of the test set . . . . .	122
5.12	General overview of the LLE-LLD-RF . . . . .	123
5.13	Training and Cross-Validation learning curves of third approach LLE-LLD-RF . . . .	124

## LIST OF FIGURES

---

5.14	The performance evaluation of the LLE-LLD-RF approach includes a ROC curve for the predicted probabilities and a confusion matrix for the test set. The ROC curve is shown in (a), while the confusion matrix is presented in (b) . . . . .	125
5.15	ROC curves comparison for the three approaches . . . . .	127
5.16	3D scatter plots for the three best features, ranked by Random Forest importance, selected from original (a), latent space (b), and LLE features (c) . . . . .	132
5.17	Density distribution plots of the top 5 features, ranked in descending order of Random Forest importance, for the original, latent space, and LLE features. The plots illustrate the distribution of feature values using KDE for the Non-Likely-COVID-19 and Likely-COVID-19 classes. . . . .	133
1	Overview of the proposed end-to-end diagnosis system . . . . .	156
2	Mobile app home and recording screens . . . . .	157
3	Recording in progress and ready screens . . . . .	158
4	Request processing and error screens based on the server's response . . . . .	159
5	Model's prediction response with Likely-COVID-19 and Non-Likely-COVID-19 . . . .	159

## ACRONYMS

**AdaBoost** Adaptive Boosting

**ADASYN** Adaptive Synthetic

**AI** Artificial Intelligence

**ANN** Artificial Neural Network

**APC** Auto regressive Predictive Coding

**BCE** Binary Cross Entropy

**BiGRU** Bidirectional Gated Recurrent Unit

**BiLSTM** Bidirectional Long Short-Term Memory

**BN** Batch Normalization

**CBAM** Convolutional Block Attention Module

**cDNA** complementary DNA

**CNN** Convolutional Neural Network

**COPD** Chronic Obstructive Pulmonary Disease

**CT** Computed Tomography

**CV** Computer Vision

**Cross-Validation** Cross-validation

**DA** Discriminant Analysis

**DCNN** Deep Convolutional Neural Network

**DL** Deep Learning

**DNN** Deep Neural Network

## ACRONYMS

---

**ECG** Electrocardiogram

**Extra-Trees** Extremely Randomized Trees

**FFT** Fast Fourier Transform

**FT** Fourier Transform

**GB** Gradient Boosting

**GD** Gradient Descent

**GFCCs** Gamma-tone Frequency Cepstral Coefficients

**GI** Gini index

**GRU** Gated Recurrent Unit

**KDE** Kernel Density Estimation

**kNN** k-Nearest Neighbors

**LDA** Linear Discriminant Analysis

**LLD** Low-Level Descriptors

**LLE** Locally Linear Embedding

**LOOCV** Leave-one-out Cross-validation

**LPC** Linear Predictor Coefficients

**LSTM** Long-Short Term Memory

**MFCCs** Mel-Frequency Cepstral Coefficients

**ML** Machine Learning

**MLP** Multi-Layer Perceptron

**MSE** Mean Squared Error

**NLP** Natural Language Processing

**NPRE** Nasal pressure

**OSA** Obstructive Sleep Apnea

**PCA** Principal Component Analysis



---

**PCR** Polymerase Chain Reaction

**PLSR** Partial Least-Squares Regression

**PSG** Polysomnography

**ReLU** Rectified Linear Unit

**RFE** Recursive Feature Elimination

**RMSE** Root Mean Square Error

**RNN** Recurrent Neural Network

**ROC** Receiver Operating Characteristic

**ROC-AUC** Receiver Operating Characteristic Area Under Curve

**RT-PCR** Reverse Transcription Polymerase Chain Reaction

**SAE** Stacked AutoEncoder

**SARS-CoV2** Severe Acute Respiratory Syndrome CoronaVirus 2

**SDB** Sleep-Disordered Breathing

**SE** Squeeze-and-Excitation

**SFS** Sequential Forward Selection

**SGD** Stochastic Gradient Descent

**SHHS** Sleep Heart Health Study

**SLP** Single-Layer perceptron

**SMOTE** Synthetic Minority Oversampling TEchnique

**SR** Speech Recognition

**STFT** Short-Time Fourier Transform

**SVD** Singular Value Decomposition

**SVM** Support Vector Machine

**t-SNE** t-distributed Stochastic Neighbourhood Embedding

**TanH** Hyperbolic Tangent

## ACRONYMS

---

**TB** Tuberculosis

**UAR** Unweighted Average Recall

**UC** University of Cambridge

**UL** University of Lleida

**WST** Wavelet Scattering Transformation

**XGBoost** Extreme Gradient Boosting

**ZCR** Zero Crossing Rate

## LIST OF ALGORITHMS

1	k-Fold Cross-Validation [7]	15
2	GridSearch	16
3	Random Forest training algorithm	22
4	Backpropagation algorithm for ANN training [102]	27
5	SMOTE Algorithm [23]	45



## INTRODUCTION

The field of medical diagnosis has always been of great significance, as an accurate and timely diagnosis can be critical in ensuring the best possible results for patients, thus, it has been always a center of interest to the research community.

As different types of data become more available, and Artificial Intelligence (AI)-based techniques and algorithms become more developed and efficient, the use of ML and DL in medical diagnosis has largely grown. These techniques have shown great promise in improving the accuracy and efficiency of medical diagnosis, and they have been widely applied in diagnostic tasks, such as clinical information diagnoses, image diagnoses, and audio diagnoses.

## 1.1 COVID-19 outbreak

By December 2019, from Wuhan, China, the world faced a new disease from the Severe Acute Respiratory Syndrome CoronaVirus 2 (SARS-CoV2) COVID-19. This disease is declared a global pandemic because of its quick spread, the cause of death of more than six million, and infection of more than 600 million people across the world, as of February 2023 [143, 145]. As above-mentioned, COVID-19 is spreading quickly because of its possible air viral transmission, in a form of aerosols, it was proved in a study on the stability of the surface [133], that the SARS-CoV-2 was stable on its surface for about three hours. Moreover, the SARS-CoV-2 can be traveling through the generated droplets from sneezing, speaking, or coughing, making a risk of contagion if the inter-person distance is less than 30 feet [108], and according to the same study, wearing a face mask can safely reduce the inter-person distance to 6.5 feet. In order to limit the SARS-CoV-2 virus from spreading quickly, there has been a global need for information and knowledge about SARS-CoV-2 diagnostic and surveillance technology. Healthcare professionals may select where and how to concentrate efforts and resources to properly isolate and treat affected people, with

timely and precise in-place diagnoses. This process can slow down the mortality rate and delay the spread of the virus. In order to lessen the devastation caused by the disease and stop its spread, frontline healthcare professionals and authorities may greatly benefit from all of the technologies needed for the quick identification of COVID-19. Several techniques have been employed globally to recognize and diagnose COVID-19 since its emergence. The experience gained from the previous SARS outbreak proved to be useful, the previously developed diagnostic tools for SARS-CoV-1 were adapted and improved to help identify the new virus, SARS-CoV-2, within a few days. Specifically, there were several techniques to screen and identify SARS-CoV-2 such as Transmission Electron Microscopy which is used to visualize the virus's morphology[97], Full Genome sequencing which played a crucial role in identifying the virus and developing specific primers and probes for Polymerase Chain Reaction testing [62], and CT imaging-based methods [28].

## 1.2 Testing and diagnosing tools

### Reverse Transcription Polymerase Chain Reaction (RT-PCR)

RT-PCR technique has been widely used in viral detection, diagnostics, and other research fields. It has been used extensively in the detection of SARS-CoV-2, the virus responsible for the COVID-19 pandemic [27]. More precisely, RT-PCR presents a molecular biology technique used to amplify RNA sequences by converting them to complementary DNA (cDNA) via reverse transcription and then amplifying the resulting cDNA using Polymerase Chain Reaction (PCR). Unlike other molecular tests which do not have good specificity, RT-PCR was considered the gold standard for COVID-19 diagnosis tools [126]. Regarding the process, the application of RT-PCR consists of several steps [35], including:

1. RNA Extraction: A patient's sample, such as a nose swab, blood, or tissue, is utilized to extract RNA using an appropriate RNA isolation kit or methodology.
2. Reverse Transcription: Using a reverse transcriptase enzyme and primers that are particular to the desired RNA target, the extracted RNA is then reverse-transcribed to create cDNA.
3. cDNA Amplification: A DNA polymerase enzyme and gene-specific primers are employed in PCR to amplify the resultant cDNA. To guarantee effective amplification, ideal cycle parameters, including temperature and time, are selected.
4. Detection: Many techniques, such as gel electrophoresis, fluorescent dyes, or probes designed specifically for the amplified product, are used to find the amplified PCR products.

### **Rapid Antigen Detection Test**

Rapid Antigen Detection is a diagnostic test for COVID-19 that works by detecting certain viral proteins in respiratory samples, unlike RT-PCR tests, which detect the genetic material of the virus, rapid antigen tests detect viral proteins directly from a patient by taking a nasopharyngeal swab or an anterior nasal swab, and the results may be acquired in few minutes, making it a faster and more practical choice for COVID-19 testing [61]. Although, it's crucial to remember that antigen testing often has lower sensitivity than RT-PCR tests and a higher likelihood of returning false-negative [43]. Positive antigen test results are therefore thought to be quite reliable, but negative ones should have verification by an RT-PCR, especially in those who have symptoms or have been exposed to COVID-19 before [83]. There are a few steps to perform a rapid antigen test:

1. Collection: A patient's sample, such as a nose swab, blood, or tissue, is collected.
2. Preparation: The sample is mixed with substances such as detergents or enzymes that can help uncover the viral particles from possible existing protective layers of proteins, lipids, or other substances, and this, allows antibodies on the test device to interact and detect viral proteins.
3. Application: The processed sample is introduced to a test device that contains antibodies able to identify the viral proteins.
4. Incubation: Giving the antibodies the time to react with the viral proteins in the sample. The test device is incubated for a brief duration, typically 10 to 15 minutes.

### **1.3 ML in COVID-19 Diagnosis**

The capacity of AI to efficiently and precisely evaluate massive volumes of data has made it a useful weapon in the fight against COVID-19. Predictive models for the diagnosis of COVID-19 are being developed using AI-based approaches including ML, DL, and Computer Vision (CV), enabling healthcare professionals to make quick and more informed decisions regarding patient treatment. Clinical data under COVID-19 are being analyzed using ML in order to find anomalies and monitor the course of the illness [5, 40, 42, 60, 114, 152]. In the context of CV, chest X-rays, and CT scans as well, have been widely used for diagnosing COVID-19, several studies have shown that COVID-19 pneumonia can cause characteristic changes in the lungs that can be detected through medical imaging, including ground-glass opacities, consolidation, crazy-paving pattern, and peripheral distribution of lesions. [12, 90]. Chest X-rays and CT scans have been shown to be effective in identifying COVID-19 pneumonia, with some studies reporting high sensitivity and specificity using different DL approaches [13, 34, 36, 68, 89, 138]. Regarding audio analysis, a significant challenge was raised by some researchers to use AI-based solutions to

quickly, safely, and efficiently diagnose COVID-19 from coughs, or respiratory sounds, due to the lack of data, the variability of cough sounds, and the difficulty in handling and analyzing audio data. However, recent studies have made strides in tackling this challenge, by first, enlarging existing datasets and making them available for research purposes [19, 22, 87, 91] then exploring and utilizing several ML and DL techniques and approaches to perform efficient diagnosis systems [3, 19, 55, 92]. In terms of COVID-19 detection accuracy, techniques like X-ray and CT scan medical analysis can produce good diagnosis results, occasionally even outperforming RT-PCR. Nevertheless, because patients have to be present physically for these procedures, the risk of a further virus spreading is increased. This emphasizes the value of a contactless analysis, and that is considered one of the motivating factors for the work of this thesis.

## 1.4 COVID-19 Symptoms

The COVID-19 symptoms can be minor to severe, and some people may even have no symptoms at all. Depending on the virus variant, COVID-19 symptoms often show up 2 to 14 days after exposure to the virus [150]. According to the WHO [143], COVID-19 symptoms can be categorized into three groups:

- The most frequently reported ones that include fever, cough, fatigue, and loss of taste or smell.
- Some less common symptoms include sore throat, headache, muscle pain, diarrhea, skin rash, discoloration of fingers or toes, and red or irritated eyes.
- Certain symptoms that can be particularly severe and require immediate medical attention, such as difficulty breathing or shortness of breath, loss of speech or mobility, confusion, and chest pain.

In order to define the main symptoms of the disease, a meta-analysis study on 54 published works that include clinical data i.e. the presence of symptoms or questionnaires has been done [6] where a total of 6382 samples were included in the statistical analysis. 6380 in the 54 studies have recorded cough symptom, whereas 5298 in 53 studies have recorded fever symptom. Dyspnea (shortness of breath) has been recorded in 3388 in only 27 studies, and 3803 from 22 studies denoted for the fatigue symptom. Thus, the mentioned study defines and confirms the most 3 common symptoms of COVID-19; Cough, Fatigue, and Shortness of breath.

## 1.5 Motivations

What motivates our research? The COVID-19 pandemic has highlighted the need for accurate, efficient, and more precisely, safe diagnostic tools to combat the spread of the virus. The development of such tools has become a top priority for healthcare professionals worldwide. Integrating AI



approaches in medical diagnosis, particularly in the analysis of cough sounds, offers a promising solution as it will give the possibility to make diagnostics remotely. This thesis work aims to investigate the integration of AI approaches in medical diagnosis, with a focus on analyzing cough sounds as a potential indicator of COVID-19 infection. The following motivations have inspired this research and driven the investigation being undertaken:

1. The timely release of RT-PCR test results is a critical factor in containing the spread of COVID-19. However, in some countries, the process of obtaining and releasing these test results may take hours or even days, leading to delays in diagnosis, and treatment. Without isolation, this can possibly increase the risk of virus transmission.
2. The availability of RT-PCR test kits is a major concern in some countries, where the quantity of test kits may be insufficient or lacking due to economic constraints. This shortage of testing resources has limited the ability of healthcare systems to detect, diagnose, and contain the spread of COVID-19.
3. As mentioned in the COVID-19 symptoms section, and reported in a previous study [40], the cough symptom had the highest correlation with positive COVID-19 cases.
4. Coughing is one of the symptoms of many chest and respiratory diseases such as Asthma, Bronchitis, Tuberculosis, and Pertussis, and as we previously discussed, it is additionally one of the most common symptoms of COVID-19 illness. To differentiate coughs linked to COVID-19 from those caused by other respiratory illnesses, more investigation is necessary. An earlier study [129] presents a method for analyzing cough sounds quantitatively, which is capable of characterizing significant differences in the sound after a Methacholine challenge. The analysis suggests that acoustic examination of cough sounds may provide information on airway flow mechanics during cough, potentially aiding in the diagnosis of respiratory disorders. Another study [21] focuses on distinguishing between dry and wet cough sounds. To achieve this, the authors suggest using two features: the number of peaks of the energy envelope and the power ratio of two frequency bands. They tested their approach on a set of eight highly dry and eight highly wet cough sound recordings and observed a clear differentiation between the two types of cough sounds using these features. The aforementioned research has shown that coughing has the potential to differentiate between COVID-19-related coughs and those associated with other diseases. However, it is important to know that there are several respiratory and non-respiratory medical conditions that can also cause coughing [55], which can create challenges that may require more attention.
5. Access to large and diverse datasets is crucial for the development of accurate and reliable models in medical diagnosis. In this work, our research has utilized the largest publicly available dataset of cough sounds under COVID-19, COUGHVID [91], it provides more than

20,000 recordings which present a valuable resource for the development of an effective AI-assisted diagnosis tool. However, the dataset suffers from class imbalance issue, which requires more care in this respect.

6. The need for a remote, safe, quick, efficient, and accurate diagnostic tool has become crucial in the context of the COVID-19 pandemic to avoid physical contact between patients and healthcare providers and reduce the spread of the virus.

## 1.6 Thesis objectives and research questions

The overarching objective of this thesis is to develop and evaluate the performance of an AI-assisted diagnosis system for COVID-19 based solely on cough sound data. Additionally, we also have some sub-objectives listed below:

- This thesis will explore the theoretical background of the ML, DL, data augmentation, and dimensionality reduction techniques used in the development of an AI-assisted diagnosis system for COVID-19 using cough sound data. By providing a detailed examination of the theoretical foundations of these methods, the thesis aims to contribute to the broader theoretical understanding of these techniques and their potential applications in various domains.
- In addition to exploring several studies that attempted to diagnose respiratory diseases using ML and DL, we present a succinct and critical review of related works regarding the diagnosis of COVID-19 utilizing cough and respiratory sounds using ML and DL approaches as well as the existing datasets.
- We analyze the COUGHVID dataset and propose a raw signal, spectral, tabular data augmentation, and class balancing to create more variability in a way to enhance the efficiency of ML and DL-based solutions.
- The present study proposes a new framework for COVID-19 diagnosis that relies solely on cough sound data as input. The framework is based on a hybrid DL approach that incorporates an attention-based CNN and LSTM. This approach is designed to effectively recognize and distinguish between positive and negative COVID-19 cases.
- The current study aims also, to propose a novel ensemble learning-based framework for COVID-19 diagnosis that utilizes Random Forest as the primary classifier. The proposed approach involves training the model on a large set of low-level audio features extracted from cough sounds. The approach has been enhanced through two additional methods: first, by incorporating a Stacked Autoencoder to extract the most discriminant features from the large set of audio features, and second, by utilizing LLE to reduce the feature space and capture more important patterns in the data.

The following research questions were developed to guide this study and achieve its objectives:

1. Can ML models trained on cough sound data accurately distinguish between positive and negative cases of COVID-19?
2. Can ML methods achieve comparable performance to DL methods in developing accurate and efficient diagnosis systems for COVID-19 based on cough sound data, or is DL necessary to achieve the desired level of performance?
3. How can the issue of class imbalance (the lack of positive samples) in COVID-19 cough sound datasets be addressed to improve the performance of AI-assisted diagnosis systems?

## **1.7 Thesis outline**

The thesis is divided into two main parts. The first part will provide an overview of the theoretical background of ML, DL, data augmentation, and dimensionality reduction techniques used in COVID-19 diagnosis from cough sound as well as a detailed and concise literature review of the field. The second part will focus on our contributions, which aim to diagnose COVID-19 cough sounds using various ML, and DL techniques.

### **Part I: Theoretical background and literature review**

#### **Chapter II — Theoretical Background:**

Provides an in-depth discussion of ML and DL concepts, as well as the methods used in the contributions of this thesis. This chapter aims to establish a strong foundation for the rest of the thesis and provide a comprehensive understanding of the relevant used theories and techniques, as well as an exploration of the state-of-the-art of ML and DL for respiratory diseases diagnostic.

#### **Chapter III — Literature Review:**

Focuses on the data availability, the state-of-the-art and an in-depth literature review of ML and DL methods for diagnosing COVID-19 from cough sound.

### **Part II: Our approaches for diagnosing COVID-19 from cough sound**

#### **Chapter IV — Hybrid CNN-LSTM with Attention-mechanism architecture and SpecAugment for COVID-19 diagnosis from Cough sound:**

This chapter provides a detailed description of the proposed framework which consists of applying SpecAugment technique for spectral data augmentation and then, hybridizing CNN and LSTM along with attention mechanism and presents the results of the conducted experiments, followed by a discussion and conclusion of the work.

**Chapter V — The application of Random Forest method for COVID-19 diagnosis from Cough Sound**

Focuses on, first, the diagnosis of COVID-19 from cough sound using a simple ensemble learning framework based on Random Forest and LLD for audio, then, the enhancement of the former by using a Stacked Autoencoder for dimensionality reduction. Lastly, the use of LLE method instead of Stacked Autoencoder to boost the performance of the classification model. The chapter provides a detailed description of each of the proposed approaches, the experimental setup, and the obtained results. Finally, a thorough discussion, ablation study and conclusion of the work are presented.

**Conclusion and perspectives for future work:**

Concludes the present work by summarizing the proposed approaches and the achievements of each one, and also illustrating some perspectives for future work and challenges regarding the field.

**1.8 Publications**

The research presented in this thesis is based on several publications, which have been listed below. These publications were either authored or co-authored by the author of this thesis, they include a journal paper, a paper under review for a journal publication, one book chapter, and two conference proceedings.

1. S. Hamdi, A. Moussaoui, M. Oussalah, M. Saidi, Early COVID-19 Diagnosis from Cough Sound Using Random Forest and Low-Level Descriptors, in: Third Int. Conf. Comput. Inf. Sci. 2021, 2021: pp. 1–6.
2. M. Berrimi, S. Hamdi, R.Y. Cherif, A. Moussaoui, M. Oussalah, M. Chabane, COVID-19 detection from Xray and CT scans using transfer learning, in: 2021 Int. Conf. Women Data Sci. Taif Univ. (WiDSTaif ), 2021: pp. 1–6. <https://doi.org/10.1109/WiDSTaif52235.2021.9430229>.
3. S. Hamdi, M. Oussalah, A. Moussaoui, M. Saidi, Attention-based hybrid CNN-LSTM and spectral data augmentation for COVID-19 diagnosis from cough sound, *J. Intell. Inf. Syst.* (2022). <https://doi.org/10.1007/s10844-022-00707-7>.
4. S. Hamdi, A. Moussaoui, M. Oussalah, M. Saidi, Autoencoders and Ensemble-Based Solution for COVID-19 Diagnosis from Cough Sound BT - Modelling and Implementation of Complex Systems, in: S. Chikhi, G. Diaz-Descalzo, A. Amine, A. Chaoui, D.E. Saidouni, M.K. Kholadi (Eds.), Springer International Publishing, Cham, 2022: pp. 279–291.

5. S. Hamdi, A. Moussaoui, M. Oussalah, M. Saidi, Locally Linear Embedding and Ensemble learning for COVID-19 Screening from Cough Sound, *Electronics Letters* (2023) Under Review.

# **Part I**

## **Theoretical Background and Literature Review**

## THEORETICAL BACKGROUND

### 2.1 Introduction

The field of ML has fundamentally transformed the approach to problem-solving across various fields. In simple words, ML consists of training an algorithm to learn from data to perform a particular task. The importance of data in this process cannot be overstated, as it provides algorithms with the knowledge necessary to enhance their performance. In this chapter, we provide an overview of the fundamental concepts and techniques that form the basis of our work on diagnosing COVID-19 from cough sound. We start by introducing ML's essential types, algorithms, and techniques, including dimensionality reduction, data augmentation, and ensemble learning. We then, delve into DL methods that have been applied to our work, including CNN, LSTM, and attention mechanism. Throughout this chapter, we also explore the various types of audio features utilized in our research, such as LLD, spectrograms, and mel-spectrograms. Lastly, we will discuss some recent works that have investigated respiratory diseases using ML and DL. By the end of this chapter, readers will have a comprehensive understanding of the theoretical background necessary to comprehend our approaches for COVID-19 diagnosis from cough sound. This includes a general overview of ML and a detailed understanding of the specific methods and techniques used in the present work.

### 2.2 Introduction to Machine Learning

ML can be defined as a subfield of AI that trains algorithms to automatically learn patterns from data and make predictions or decisions for unseen instances without being explicitly programmed. It aims to enable computers to learn from data and improve their performance on a particular task, without being explicitly programmed for that task. In essence, ML algorithms use statistical

techniques as well as optimization algorithms that aim to fit a model to identify patterns in data and generalize them to make new predictions.

### 2.2.1 Types of Machine Learning

There are various types of machine learning: *supervised learning*, *unsupervised learning*, *semi-supervised learning*, and *reinforcement learning* [7].

1. **Supervised learning:** A supervised learning algorithm is trained on labeled data, which means that the input data has been prior annotated with the intended output label. Depending on the used mechanism of each algorithm, generally, reducing the error between the predicted output and the actual output is performed, the algorithm learns to map the input data to the proper output. Supervised learning is used in tasks such as classification and regression. The classification consists of predicting the category or class, such as predicting whether a cough sample belongs to a COVID-19-positive or a healthy individual. On the other hand, regression refers to the learning process that aims to predict a continuous output, such as predicting a patient's blood glucose levels based on factors such as their age, weight, and family history of diabetes. This can help physicians better manage and treat patients with diabetes.
2. **Unsupervised learning:** Unsupervised learning involves processes that aim to discover underlying patterns in data without relying on labeled examples. Unsupervised learning is used commonly for clustering and dimensionality reduction. The clustering is based on grouping data points, the algorithm finds patterns or resemblances in the data without knowing what these patterns could be, whereas the dimensionality reduction involves reducing the number of features in a dataset. Its main aim is to pinpoint the most important and instructive features that can explain the variance of data. Various algorithms and techniques are used for dimensionality reduction including Locally Linear Embedding (LLE), and Autoencoders which will be discussed in Section 2.8.
3. **Semi-supervised learning:** Combines labeled and unlabeled data to improve model accuracy, especially when obtaining labeled data is difficult or costly. By using both types of data, algorithms can better identify patterns and achieve better performance than supervised learning alone. This approach is useful in medical diagnosis tasks and can enhance ML model performance.
4. **Reinforcement learning** Involves an algorithm interacting with an environment and receiving rewards or punishments to learn the optimal action to take for maximum reward. It's commonly used in robotics and can also be applied to some medical diagnosis tasks. Techniques like Deep Q-Network and Policy Gradient methods are used for reinforcement learning.



### 2.2.2 Machine learning pipeline

ML pipeline framework guides the general process of creating a ML model, from data acquisition to model evaluation. It provides a systematic approach to ensure accurate and efficient model development. This section will cover the essential elements of the machine learning pipeline and their best practices.

1. **Data collection and preparation:** The first stage of the ML pipeline involves collecting and preparing data, including identifying sources, cleaning, and preprocessing. Data quality is crucial as it impacts model performance, and the type of data determines the appropriate model or technique to use. This step also involves exploratory data analysis and splitting data into training, validation, and test sets.
2. **Data exploration and visualization:** This step involves performing statistical analysis and visualization to identify patterns and correlations between features using techniques like scatter plots, histograms, and heat maps. It is essential to resolve data quality issues such as missing values and outliers before model training.
3. **Feature engineering:** Involves, in general, feature extraction and feature selection. Feature extraction transforms raw data into a format suitable for ML algorithms, Mel-Frequency Cepstral Coefficients (MFCCs) in audio data for instance, while feature selection selects the most relevant subset of features to avoid redundancy and overfitting. For example, RFE is a common method used for feature selection, which recursively removes features from the dataset and builds a model on the remaining features until the desired number of features is reached.
4. **Model selection, data split, and training:** In this step, the appropriate machine learning algorithm is selected based on the problem, data type, and desired performance, which may require experimentation with several algorithms. The data is split into training and testing subsets, with a commonly used split of 70-80% for training and 20-30% for testing, and the testing set should be independent of the training set. The training phase involves fitting the algorithm to the data and adjusting parameters or weights to minimize the error between real and predicted outputs. In clustering, the algorithm aims to group similar data points together and separate dissimilar points based on the maximum distance between classes and the minimum distance within each class.
5. **Model evaluation:** The performance of a trained model is determined to show its ability in predicting outcomes on unseen data. Different metrics are used to evaluate the model, including *Accuracy*, *Precision*, *Sensitivity*, *F1-score*, *ROC-AUC*, *Specificity*, and others, depending on the problem and the model type. This process helps identify the strengths and weaknesses of the model and guides the selection of a more appropriate model or adjustments to its hyper-parameters. We can list the above-mentioned evaluation metrics

which they have been used in this work. In the following formulations,  $TP$ ,  $TN$ ,  $FP$ , and  $FN$ , refer to the component of the confusion matrix, true positive, true negative, false positive, and false negative.

- Accuracy is the ratio of the number of correctly classified samples to the total number of samples and can be represented mathematically as follows:

$$(2.1) \quad Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision is a metric that measures the proportion of correctly predicted positive instances out of all predicted positive instances. It can be represented with the following formula:

$$(2.2) \quad Precision = \frac{TP}{TP + FP}$$

- Sensitivity, recall, or true positive rate, measures the proportion of actual positive cases that are correctly predicted by the model. It can be calculated as:

$$(2.3) \quad Sensitivity = \frac{TP}{TP + FN}$$

- F1-score which combines precision and recall into a single score, and it is calculated as the harmonic mean of precision and recall:

$$(2.4) \quad F1 = 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$$

- ROC-AUC is commonly used to measure the performance of binary classification models, which measures the area under the Receiver Operating Characteristic (ROC) curve. In other words, A higher ROC-AUC score indicates a better performance of the model in distinguishing between the two classes. The ROC curve plots the Sensitivity against the false positive rate (FPR) at various threshold values. Mathematically, ROC-AUC can be formulated as:

$$(2.5) \quad ROC - AUC = \int_0^1 Sensitivity(FPR^{-1}(t))dt$$

where  $FPR^{-1}(t)$  is the inverse function of the FPR at threshold  $t$ .

- Specificity, also known as true negative rate, is the proportion of the negatives that are correctly predicted as negatives by the model. Mathematically, it can be computed as:

$$(2.6) \quad Specificity = \frac{TN}{TN + FP}$$

6. **Model validation:** In the ML pipeline, the validation step is critical to assess the performance of the trained model on new data. To achieve this, model validation techniques such as  $k$ -fold Cross-Validation and Leave-one-out Cross-validation (LOOCV) are used. In our contributions, we used  $k$ -fold Cross-Validation, in which, the dataset is divided into  $k$  subsets, and the model is trained  $k$  times, each time,  $k - 1$  subsets are used for training and the remaining subset for validation. After computing the intended evaluation metrics  $k$  times, the average of each metric is finally considered. Algorithm 1 describes the process of  $k$ -fold Cross-Validation.

---

**Algorithm 1**  $k$ -Fold Cross-Validation [7]

---

**Require:**  $X$ : training data,  $k$ : number of folds

**Ensure:** average performance metric of the model

- 1: Divide  $X$  into  $k$  equal-sized folds
  - 2: **for**  $i \leftarrow 1$  to  $k$  **do**
  - 3:    $X_{test} \leftarrow i^{th}$  fold
  - 4:    $X_{train} \leftarrow X$  with  $X_{test}$  removed
  - 5:   Train the model on  $X_{train}$
  - 6:   Test the model on  $X_{test}$
  - 7:   Calculate performance metric on  $X_{test}$
  - 8:   Add performance metric to list
  - 9: **end for**
  - 10: Calculate the average of the performance metrics
- 

7. **Hyper-parameter tuning:** Hyper-parameters are parameters that control the behavior of the learning algorithm, and their values need to be chosen carefully to achieve optimal performance. There are various techniques available for tuning hyper-parameters, such as GridSearch, random search, and Bayesian optimization. In this process, the model is trained and validated using different sets of hyper-parameters, and the best set of hyper-parameters is selected based on the evaluation metric. In our contributions, we used GridSearch to tune hyper-parameters, which involves exhaustively searching through a predefined set of hyper-parameters and selecting the set that produces the best performance. Algorithm 2 describes the process of GridSearch.

Figure 2.1 illustrates a simplified overview of the ML pipeline, from data acquisition and to model training and prediction.

## 2.3 Common machine learning algorithms

This section will detail some of the most widely used ML algorithms. These algorithms include linear and polynomial regression, SVM for supervised learning, and  $k$ -means clustering algorithm for unsupervised learning.

**Algorithm 2** GridSearch

**Require:**  $X$ : training data,  $y$ : target labels, *parameters*: dictionary of hyperparameters with possible values,  $k$ : number of folds for cross-validation

**Ensure:** best hyperparameters for the model

- 1: Define a list to hold all the possible combinations of hyperparameters
- 2: Initialize the performance metric to track the best model so far
- 3: **for** each possible combination of hyperparameters **do**
- 4:   Train the model on  $X$  with the given hyperparameters
- 5:   Evaluate the model using  $k$ -fold cross-validation and the chosen performance metric
- 6:   **if** the model's performance is better than the previous best **then**
- 7:     Update the performance metric
- 8:     Save the hyperparameters that resulted in the better performance
- 9:   **end if**
- 10: **end for**
- 11: **return** the hyperparameters that resulted in the best performance

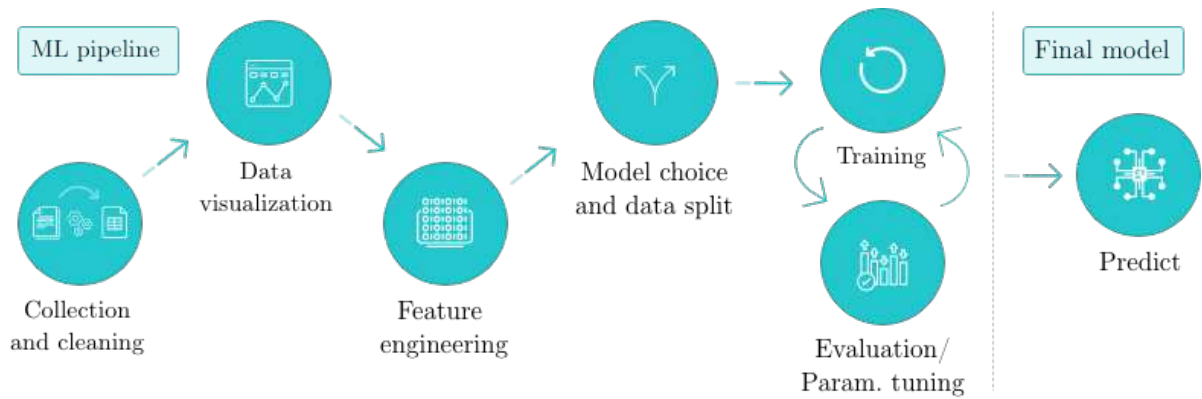


Figure 2.1: Overview of the ML Pipeline, showing the various stages, from data collection to model prediction

### 2.3.1 Linear and polynomial regression

In supervised learning, linear regression is a widely used technique for predicting a continuous outcome variable based on one or more input variables which are features. Linear regression aims to find the best linear relationship between the input variables and the target variable [104]. Linear regression assumes that the relationship between the input variables  $x_1, x_2, \dots, x_n$  and the outcome variable  $y$  can be represented by a linear equation of the form:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n + \epsilon$$

where  $\theta_0, \theta_1, \dots, \theta_n$  are the coefficients or parameters of the model, and  $\epsilon$  is the error term that captures the difference between the predicted and actual values of  $y$ . The objective of linear regression is to estimate the values of  $\theta_0, \theta_1, \dots, \theta_n$  that best fit the observed data. This is typically done by minimizing the loss function Mean Squared Error (MSE) between the predicted values

$\hat{y}_i$  and the actual values  $y_i$  for all  $d$  observations in the training dataset:

$$MSE = \frac{1}{d} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

where  $d$  is the number of samples. Minimizing MSE depends on the values of  $\theta_0, \theta_1, \dots, \theta_n$ , which can be estimated using various algorithms, the most commonly used is Gradient Descent (GD). Polynomial regression is an extension of linear regression that models a non-linear relationship between the independent variable  $x$  and the dependent variable  $y$  as an  $n$ th-degree polynomial function. The equation for the polynomial regression model with degree  $n$  can be represented as:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots + \theta_n x^n + \epsilon$$

### 2.3.2 Support Vector Machine (SVM)

SVM is one of the most well-known supervised algorithms, The main idea behind SVM is to find the optimal linear separating hyperplane i.e. the decision boundary that separates the data of one class from another [17]. Consider a dataset  $D$  be given as  $(X_1, y_1), (X_2, y_2), \dots, (X_{|D|}, y_{|D|})$  where  $X_i$  is a training instance associated to a class label  $y_i$  where  $y_i \in \{+1, -1\}$ , and the data are linearly separable.

$$(2.7) \quad \begin{aligned} w \cdot x_i + b &\geq +1 \text{ where } y_i = +1 \\ w \cdot x_i + b &\leq -1 \text{ where } y_i = -1 \end{aligned}$$

where  $w$  is a weight vector  $W = \{w_1, w_2, \dots, w_n\}$ ;  $n$  is the number of features,  $b$  is a bias. By multiplying each equation by the associated label  $y_i$  we get:

$$(2.8) \quad y_i(w \cdot x_i + b) \geq +1$$

The optimal hyperplane is:

$$(2.9) \quad w_0 \cdot x + b_0 = 0$$

By using equations 2.7, we can obtain the width of the margin defined by only the **Support Vectors**, which are the training samples that satisfy:

$$(2.10) \quad y_i(w \cdot x_i + b) = 1$$

Then, the width of the margin can be denoted by:

$$(2.11) \quad \rho(w_0, b_0) = \frac{2}{\|w\|}$$

The objective is to maximize Equation 2.11 and so minimize  $\frac{1}{2}\|w\|^2$ , which can be solved by quadratic programming.

$$(2.12) \quad \begin{aligned} \min_w \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w \cdot x_i + b) \geq 1 \end{aligned}$$

Using Lagrangian multipliers  $\mathbf{L}(\mathbf{w}, \mathbf{b}, \alpha)$  and the partial derivatives of  $\mathbf{L}$ , we get the dual of the problem 2.12:

$$(2.13) \quad \begin{aligned} \max \quad & \sum_{i=1}^{|D|} \alpha_i - \sum_{i=1}^{|D|} \sum_{j=1}^{|D|} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ \text{s.t.} \quad & \sum_{i=1}^{|D|} \alpha_i y_i = 0 \end{aligned}$$

Furthermore, in the case of non-linearly separable data, penalizing misclassification should be put forward using *Cost factor*  $\mathbf{C}$ , and *slack-variables*  $\xi_i$ , thus, the optimization problem 2.12 becomes:

$$(2.14) \quad \begin{aligned} \min_w \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{|D|} \xi_i \\ \text{s.t.} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \forall i \in \{1, 2, \dots, |D|\} \end{aligned}$$

Hence, this formulation 2.14 introduces the *Soft margin* concept, which makes a compromise between a **large margin** and **less misclassification**. The parameter  $C > 0$  controls the effect of the penalization. This parameter is often used for regularization to reduce overfitting.

SVM utilizes a Kernel Method where a non-linear mapping is applied to transform the input data to a higher dimensional space, allowing SVM to find a linear separation in this new space. The kernel mapping can be represented as follows:

$$(2.15) \quad K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$$

Various kernels have been used to map the data to higher dimensional space, including [96]:

- Polynomial Kernel of degree  $h$ :  $K(X_i, X_j) = (X_i \cdot X_j + 1)^h$
- Gaussian radial basis function kernel:  $K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$
- Sigmoid kernel:  $K(X_i, X_j) = \tanh(kX_i \cdot X_j - \delta)$

### 2.3.3 Decision Trees

Decision Tree is a popular supervised learning algorithm that is widely used for classification and regression. A decision tree is a hierarchical model that partitions the data into smaller subsets based on a sequence of binary decisions on features. Each internal node of the tree represents a decision rule based on a specific feature, and each leaf node represents a predicted outcome. The algorithm uses a greedy approach to recursively split the data into smaller subsets based on the feature that results in the maximum information gain or reduction in impurity [46]. Let  $X$  be the

input data,  $Y$  be the target variable, and  $D$  be the dataset. The goal of a decision tree algorithm is to learn a tree,  $T$ , that can predict the value of  $Y$  for any given  $X$ . The algorithm starts by selecting the feature that results in the maximum information gain or reduction in impurity. The data is then split into two subsets based on the values of the selected feature. This process is repeated recursively for each subset until a stopping criterion is met, such as a maximum depth or minimum number of data points in a leaf node. The final prediction of the decision tree model for a new instance,  $X_{new}$ , is obtained by traversing the tree from the root to a leaf node based on the decision rules and returning the predicted value associated with that leaf node. The impurity or entropy of a node,  $N$ , is defined as:

$$H(N) = - \sum_{i=1}^K p(i) \log_2 p(i)$$

where  $K$  is the number of classes, and  $p(i)$  is the proportion of data points in  $N$  that belong to class  $i$ . The information gain of a split on feature  $F$  at node  $N$  is defined as the reduction in impurity:

$$IG(N, F) = H(N) - \sum_{i=1}^2 \frac{|N(i)|}{|N|} H(N(i))$$

where  $|N(i)|$  is the number of data points in the  $i^{th}$  subset, and  $|N|$  is the total number of data points in  $N$ . The feature that results in the maximum information gain is selected for the split.

### 2.3.4 K-means

K-means is an unsupervised learning algorithm used for clustering data points into  $k$  different clusters based on their similarity (denote that  $k$  is given as input). The algorithm starts by randomly selecting  $k$  initial centroids, one for each cluster. Then, for each data point, the algorithm assigns it to the nearest centroid based on the Euclidean distance between the point and the centroid. After all data points have been assigned, the centroids are updated as the mean of all data points assigned to that centroid. The algorithm repeats the same process and stops updating centroids when there is no change. The objective of k-means is to minimize the intra-cluster sum of squares, which is defined as the sum of the squared Euclidean distances between each data point and its assigned centroid [46]. The k-means algorithm can be formulated as follows:

Given a dataset  $x_1, x_2, \dots, x_n$ , where  $x_i \in \mathbb{R}^d$ , and the number of clusters  $k$ , the objective is to minimize the intra-cluster sum of squares:

$$\text{minimize } \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2,$$

where  $C_i$  is the set of data points assigned to centroid  $\mu_i$ .

## 2.4 Ensemble Learning

Ensemble learning is a popular and widely used ML technique, in simple words, it consists of combining multiple simple classifiers to perform a powerful classifier. In this section, we will describe the three main techniques of ensemble learning which are *Bagging*, *Boosting*, and *Stacking*, as well as the ensemble learning algorithm Random Forest that has been widely used in our work.

### 2.4.1 Bagging

Bagging, or bootstrap aggregating, is a technique that trains multiple models on bootstrap samples of the training data and aggregates their predictions using a simple voting or averaging mechanism. The main idea behind bagging is to reduce the variance of the model by introducing randomness in the data and the model itself. Bagging can be used with any base model, but it is particularly effective with high-variance models such as decision trees. Let  $D$  be the training data of size  $n$ , and let  $D_i$  be a bootstrap sample of size  $n_i$  drawn from  $D$  with replacement. The bagging algorithm trains  $m$  models, denoted by  $f_i(x)$ , on the bootstrap samples  $D_i$  [46]. The final prediction is obtained by aggregating the predictions of the individual models using a simple voting or averaging mechanism:

$$f_{\text{bagging}}(x) = \frac{1}{m} \sum_{i=1}^m f_i(x)$$

### 2.4.2 Boosting

In boosting, multiple simple models are trained sequentially, and the weights of the training data are adjusted to emphasize the misclassified examples in each iteration. Let  $D$  be the training data of size  $n$ , and let  $w_i$  be the weight of example  $i$ . The boosting algorithm trains  $m$  models, denoted by  $f_i(x)$ , on the training data weighted by  $w_i$ . The weights are updated after each iteration using a weighting function  $W(y, f_i(x))$  that depends on the true label  $y$  and the model prediction  $f_i(x)$ . The final prediction is obtained by combining the individual models using a weighted average:

$$f_{\text{boosting}}(x) = \sum_{i=1}^m \alpha_i f_i(x)$$

where  $\alpha_i$  is the weight of the  $i$ -th model, and it is determined by taking the logarithm of the ratio of the error of the  $i$ -th model to the error of a random classifier:

$$\alpha_i = \log \frac{1 - \text{error}_i}{\text{error}_i}$$

where  $\text{error}_i$  is the error of the  $i$ -th classifier, calculated as the sum of the weights of the misclassified examples [46].



### 2.4.3 Stacking

In stacking, a set of base models is trained on the training data, and their predictions are used as input features to train a higher-level model [7]. Let  $D$  be the training data of size  $n$ , and let  $f_1(x), f_2(x), \dots, f_m(x)$  be the base models. The stacking algorithm trains a higher-level model, denoted by  $g(x)$ , on the predictions of the base models:

$$g(x) = h(f_1(x), f_2(x), \dots, f_m(x))$$

where  $h$  is a function that combines the predictions of the base models. The predictions of the stacking ensemble for a new instance  $x$  are obtained by feeding  $x$  into the base models, obtaining their predictions, and then feeding the predictions into the higher-level model:

$$f_{stacking}(x) = g(f_1(x), f_2(x), \dots, f_m(x))$$

### 2.4.4 Random Forest

Random Forest is one of the most common ensemble learning algorithms, that builds multiple decision trees which are well-known for their high variance (Sections 2.3.3 and 2.4.1), and combines their outputs to improve the overall accuracy of the predictions. Each decision tree in the Random Forest is built using a different subset of the training data and a random subset of the features. The algorithm then aggregates the predictions from all the trees to obtain the final prediction [7, 16, 18, 46]. Compared to one single tree model, Random Forest has several advantages including:

- **Avoiding overfitting:** Random Forest utilizes multiple decision trees, and random subsets of the data (bagging method) and features, thus, the algorithm reduces the risk of overfitting to the training data.
- **Enhanced accuracy:** By aggregating the predictions of multiple trees, the algorithm can enhance the overall prediction accuracy.
- **Feature importance:** Random Forest can be used for feature selection or interpretation, as it can provide information on the importance of different features.

Random Forest algorithm can be summarized in the following steps:

1. Randomly select a subset of the training data (with replacement) to create a new training set.
2. Randomly select a subset of the features to use for the current tree.
3. Build a decision tree using the selected data and features.
4. Repeat steps 1-3 to create multiple decision trees.

5. To make a prediction for a new data point, run it through all the trees in the forest and aggregate their predictions.

Mathematically, let  $X = x_1, x_2, \dots, x_n$  be the set of training data points, where  $x_i$  is a  $d$ -dimensional feature vector, and  $Y = y_1, y_2, \dots, y_n$  be the corresponding set of labels. Let  $T$  be the number of trees to be built in the random forest. At each node of a decision tree, the algorithm selects a feature  $j$  and a threshold value  $s$  that optimizes a certain splitting criterion. The criterion measures the purity of the resulting subsets after the split and is typically based on:

- Gini index (GI), which is defined as:

$$\text{GI} = 1 - \sum_i (p_i)^2$$

where  $p_i$  is the proportion of the number of samples of a particular class in the set.

- Or Information gain which is measured by the Entropy. A lower entropy value indicates a more pure set. It is defined by the following formula:

$$\text{Entropy} = - \sum_{i=1}^n p_i \log_2(p_i)$$

where  $p_i$  is the proportion of the number of elements of a particular class in the set, and  $n$  is the total number of classes.

Let  $m$  be the number of features to be considered at each node of a decision tree, where  $m \ll d$  ensures diversity among the trees. Let  $p$  be the proportion of the training data to be used for each tree, where  $0 < p \leq 1$ . Algorithm 3 describes the main steps of the training process of random forest.

---

**Algorithm 3** Random Forest training algorithm

---

**Require:** Training data  $X$ , labels  $Y$ , number of trees  $T$ , number of features  $m$ , proportion of training data  $p$

**Ensure:** List of decision trees  $F = f_1, f_2, \dots, f_T$

- 1:  $F \leftarrow \emptyset$
  - 2: **for**  $t = 1$  to  $T$  **do**
  - 3:  $X_t \leftarrow$  randomly select  $p \times n$  data points from  $X$  with replacement
  - 4:  $j_t \leftarrow$  randomly select  $m$  features from the  $d$  features
  - 5:  $f_t \leftarrow$  build a decision tree on  $X_t$  using the features  $j_t$
  - 6: add  $f_t$  to  $F$
  - 7: **end for**
  - 8: **return**  $F$
- 

To make a prediction for a new data point  $x_{new}$ , random forest aggregates the predictions from all the trees:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^T f_t(x_{new})$$

where  $\hat{y}$  is the predicted label,  $f_t(x)$  is the predicted label from tree  $f_t$ , and  $T$  is the number of trees in the forest. We denote also that random forest can be used for regression tasks.

## 2.5 Introduction to Deep Learning

DL is a sub-type of ML where ANNs are used to learn complex representations of data. ANNs consist of interconnected units, which are trained to recognize patterns in data through back-propagation algorithm. ANNs are not recently developed, this concept was first introduced in the 1940s [79], and since then, they have been extensively studied and improved. However, the recent development in computing power and availability of huge amounts of data have allowed making ANNs deeper, which can learn increasingly complex features of the data. This is where the term **deep** learning comes from, as it emphasizes the ability of these networks to learn from data in different representations at each level through multiple layers of units. Each layer captures different levels of patterns, thus, more complex features are learned when network is deeper. In this section, we will discuss various topics related to DL, including different architectures, back-propagation algorithm, CNN, LSTM, and attention mechanism. We will explore the underlying principles, the fundamental concept, and the main applications of each one, and specifically, what have been used in our work.

### 2.5.1 Artificial Neural Network (ANN)

The idea behind ANNs is to simulate and mimic the structure and function of the human brain, which is composed of a huge network of neurons that communicate with each other through synapses. Neurons receive inputs from other neurons, process them, and produce an output signal that is transmitted to other neurons for further processing or to make a specific action. Similarly, ANNs are made up of units that take input from other units, process it using an **activation function**, and then, the latter either sends the output to other units or achieves the decision step. Weights and biases of a unit are modified and adjusted with the aim of enhancing the performance of the network using a process called *backpropagation*. Various architectures of ANN have been developed and used for different applications, we can summarize them by mentioning the two main ones [123, 124]:

- **Feed-forward architectures:** In this type of ANN, the information takes only one direction, from the input layer to the output. There are several feed-forward architectures, few of them include Single-Layer perceptron (SLP), Multi-Layer Perceptron (MLP), Deep Neural Network (DNN), and CNN. This type of neural network is often used in applications that do not require time-dependent or sequential data processing, in other words, where the input data can be processed independently of the order in which it is presented. For instance, CNNs are often used for image and video processing, while SLP, MLP, and DNN are mostly used for general tasks.

- **Recurrent architectures:** This type is designed for sequential data processing tasks. Specifically, Recurrent Neural Network (RNN)s have feedback connections that allow them to process sequences of variable length, and they can learn to model long-term dependencies in the input data. Several recurrent architectures have been developed since the emergence of this concept including RNN, LSTM, and Gated Recurrent Unit (GRU). Unlike Feed-forward architectures, this type is well-suited for applications that require the processing of time-dependent or sequential data. Recurrent architectures are used for specific tasks in Natural Language Processing (NLP), Speech Recognition (SR) due to their ability to capture the temporal correlations in the input data. We shall note that LSTM and GRU are specialized types of RNNs that tackle the issue of **Vanishing gradients** on long sequences.

Figure 2.2 illustrates a general structure of a DNN with three hidden layers, each layer containing 7, 7, and 6 units, respectively. The arrow represents the flow of information, each edge is associated with a weight that needs to be optimized during the training process. The output of each node is determined by its input and the corresponding weight, which is then passed through an activation function to produce an output. This output is then forwarded to the units of the next layer until the final output layer is reached.

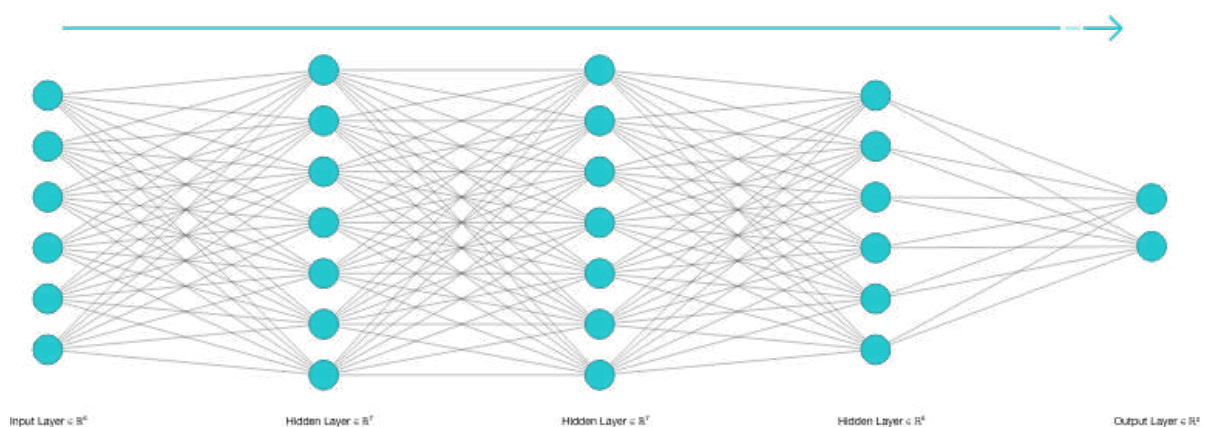


Figure 2.2: General structure of a DNN with input, output, and three hidden layers. Each unit in each layer processes input using an activation function and forwards its activation to the next layer.

The activation function is a non-linear function that introduces non-linearity into the network, allowing it to learn complex relationships between the input and output [123, 124]. Several functions have been used with neural networks, including:

- **Rectified Linear Unit (ReLU):** It applies the function  $f(x) = \max(0, x)$  to the output of a unit, where  $x$  is the input to that unit. The rectifier becomes widely used in CNN, and also in our present work.

- **Sigmoid or logistic:** This function takes any input value and maps it to a value between 0 and 1, which makes it useful for binary classification problems. It is defined as:  $f(x) = \frac{1}{1+e^{-x}}$  where  $x$  presents the input. Sigmoid has also been used in the major parts of our work.
- **Hyperbolic Tangent (TanH):** Rescaled and biased logistic function, maps the input to a value between -1 and 1 and makes each layer's output more or less normalized at the beginning of the training process. TanH is defined by  $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ . It often helps speed up convergence.
- **Softmax:** A special case of logistic, this function is mostly useful and used in the output layer of a neural network for multi-class problems. It is defined by  $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}$  for  $i = 1, \dots, k$  where  $z$  vector of real numbers  $z = (z_1, z_2, \dots, z_k)$ . It maps a  $k$ -dimensional vector of real numbers to a  $k$ -dimensional vector of real numbers between 0 and 1 that sum up to 1.

Mathematically, let's take a simple example, the classification of a suspect COVID-19 case from clinical information (50 features) as positive or negative, the formulation behind a DNN involves the computation of the output of each unit, based on its inputs, weights, and biases, using an activation function. The input to the network is denoted by  $\mathbf{x} \in \mathbb{R}^n$  where it presents the input to the first hidden layer, and the output is denoted by  $\hat{\mathbf{y}} \in \mathbb{R}^m$ , where  $n = 50$  is the dimension of the input or feature vector, and  $m = 2$  is the dimension of the output, which corresponds to positive or negative. For a given layer  $l$ , the input to a unit  $j$  is denoted by  $\mathbf{z}_j^{(l)}$  and defined as the weighted sum of the outputs from the previous layer, plus a bias term, followed by an activation function  $f$ :

$$\mathbf{z}_j^{(l)} = f \left( \sum_{i=1}^{k_{l-1}} \mathbf{w}_{ij}^{(l)} \mathbf{a}_i^{(l-1)} + b_j^{(l)} \right)$$

where  $\mathbf{w}_{ij}^{(l)}$  is the weight of the connection between the  $i$ -th neuron in the previous layer and the  $j$ -th neuron in the current layer,  $\mathbf{a}_i^{(l-1)}$  is the output of the  $i$ -th unit in the previous layer,  $b_j^{(l)}$  is the bias term for the  $j$ -th neuron in the current layer, and  $k_{l-1}$  is the number of units in the previous layer. The output of the  $l$ -th layer is given by  $\mathbf{a}^{(l)} \in \mathbb{R}^{k_l}$ , where  $k_l$  is the number of neurons in layer  $l$ . The output of the final layer is the predicted output  $\hat{\mathbf{y}}$ . The weights  $w$  and biases  $b$  are learned through a training process, which consists of minimizing a loss function that measures the difference between the predicted output  $\hat{\mathbf{y}}$  and the actual output  $\mathbf{y}$ . In our binary classification example, as well as similar problems, Binary Cross Entropy (BCE) can be used as a loss function which is defined by:

$$\mathcal{L}_{\text{BCE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^N (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

where  $N$  is the number of samples in the dataset,  $\mathbf{y}$  is the actual class vector,  $\hat{\mathbf{y}}$  is the predicted class vector. The first term in the summation penalizes the model when the true label  $y_i$  is 1 (positive) but the predicted probability  $\hat{y}_i$  is low (less likely to be positive), while the second term

penalizes the model when the true label  $y_i$  is 0 (negative) but the predicted probability  $\hat{y}_i$  is high (more likely to be positive). The loss function needs to be optimized using the backpropagation algorithm, which will be discussed in the next section.

### 2.5.2 Backpropagation algorithm

The Backpropagation algorithm is widely used for training feed-forward neural networks, where its variant Backpropagation Through Time is used to train RNN type. It is based on the gradient descent optimization technique [123]. The algorithm is used to adjust the weights of the network in order to minimize the prediction error. By sending the error backward across the network from the output layer to the input layer, the algorithm propagates errors. This is accomplished by first calculating the output layer error, which is the difference between the output that was predicted and the actual. After that, the error signal is computed, which is the derivative of the activation function multiplied by the error at the current layer, the error at each neuron in the output layer is then propagated back to the previous layer. Up until the input layer is reached, this procedure is repeated for each hidden layer of the network. Algorithm 4, shows the mechanism of the backpropagation algorithm in updating and adjusting weights using gradients.

### 2.5.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN)s are a type of neural network architecture that is widely and mainly used for image recognition tasks. CNNs are inspired by the visual cortex in the brain, which is composed of cells that are tuned to recognize specific visual stimuli in particular locations of the visual field, without taking the whole visual field into consideration. They have been initiated by Y. LeCun et al. [69] for recognizing handwritten zip codes for automated sorting operations. Specifically, this type of architecture is designed to work with images, as its basic building block is the *convolutional layer*, which consists of a set of learnable filters that slide over the input image and produce a set of feature maps by computing the dot product between the filter and the local patches or region of the input. During the training phase, the *convolutional layer* can learn from local features (particular features in an image, for instance) without taking their locations into account, thus, this can be considered as the main advantage of CNNs. CNNs are characterized by two important characteristics, compared to DNN:

- **Local connectivity:** Unlike DNNs, in which, there is a full connection between units i.e. every unit is fully connected with all units in the previous and next layers, in CNNs, each unit is connected only to a subset of the previous layer which is called *receptive field*. Consequently, the computation cost will be reduced by reducing the number of learnable parameters of the network.
- **Weight sharing:** In contrast to DNNs, where weights are typically initialized and updated independently, one by one, CNNs use shared weights that are updated jointly within each

---

**Algorithm 4** Backpropagation algorithm for ANN training [102]

---

**Require:** Training dataset  $D = (x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ , learning rate  $\eta$ , number of epochs  $num\_epochs$ , number of layers  $L$ , loss function  $J$

**Ensure:** Trained neural network weights and biases

- 1: Initialize weights and biases randomly;
- 2: **for** epoch in 1 to  $num\_epochs$  **do**
- 3:   **for** example in  $D$  **do**
- 4:     Perform forward propagation to compute the output of the last layer  $a_L$  for input  $x$ ;
- 5:     Compute error signals  $\delta_L$  for output layer;

$$\delta_L = \nabla_{\hat{y}} J \odot f'(z_L)$$

where  $\nabla_{\hat{y}} J$  is the gradient of the loss function with respect to the predicted output  $\hat{y}$ ,  $\odot$  is the element-wise multiplication, and  $f'(z_L)$  is the derivative of the activation function of the output layer with respect to its input.

- 6:   **for**  $l$  in  $L - 1$  to 1 **do**
- 7:     Compute error signals  $\delta_l$  for hidden layers using  $\delta_{l+1}$  and weights  $W_{l+1}$

$$\delta_l = (W_{l+1}^T \delta_{l+1}) \odot f'(z_l)$$

where  $W_{l+1}^T$  is the transpose of the weight matrix  $W_{l+1}$  connecting layer  $l + 1$  to layer  $l$ ,  $f'(z_l)$  is the derivative of the activation function  $f(z_l)$  evaluated at the weighted input  $z_l$  of layer  $l$ . We denote that  $\delta_L$  is the error signal for the output layer.

- 8:   **end for**
- 9:   Compute weight and bias gradients for the current example using error signals and input activations;

$$\frac{\partial J}{\partial W_{ij}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)} \quad \frac{\partial J}{\partial b_j^{(l)}} = \delta_j^{(l)}$$

where  $W_{ij}^{(l)}$  is the weight between neuron  $i$  in layer  $l - 1$  and neuron  $j$  in layer  $l$ ,  $b_j^{(l)}$  is the bias for neuron  $j$  in layer  $l$ ,  $a_i^{(l-1)}$  is the activation of neuron  $i$  in layer  $l - 1$ , and  $\delta_j^{(l)}$  is the error signal for neuron  $j$  in layer  $l$ .

- 10:   Compute the weight and bias updates using the gradient descent formulas:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \eta \frac{\partial J}{\partial W_{ij}^{(l)}} \quad b_j^{(l)} = b_j^{(l)} - \eta \frac{\partial J}{\partial b_j^{(l)}}$$

- 11:   **end for**

- 12: **end for**
-

filter or kernel. This weight sharing is possible because the filter is forced to recognize the same patterns across the entire input volume, making it more efficient in recognizing local patterns and reducing the number of parameters that need to be learned. Figure 2.3 shows the concept of shared weights and local connectivity within a convolutional layer and how the same filter is applied to different parts of the input image.

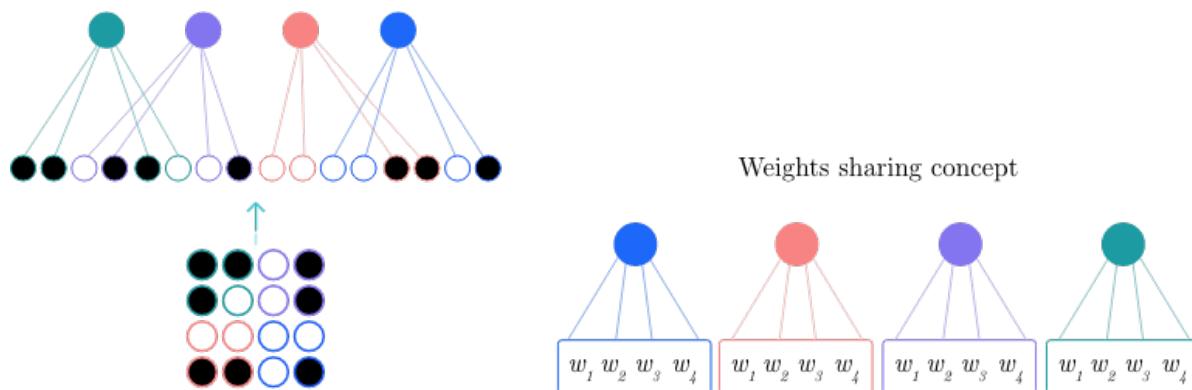


Figure 2.3: A schematic diagram of a convolutional layer that shows the receptive field of a neuron and how weights are shared within the receptive field.

Generally, a CNN is composed three main types of layers: *Convolutional layer*, *Pooling layer*, and *fully connected layers*.

- **Convolutional layer:** considered as feature extractor and the main block in CNN, uses the convolution operation to compute a set of output feature maps from the input feature map or image. A convolutional layer is a set of independent filters, each one is independently convolved with the input image to give a set of feature maps. This type of layer is *parametrized* i.e. it has a set of learnable weights which form the *filters*. Filters are generally square grids or matrices of discrete numbers, their width and height are smaller than the input image [33]. The output of the  $k$ -th feature map in the  $l$ -th convolutional layer is computed by convolution operation which is defined by:

$$z_k^{(l)} = \sum_{c=1}^{C_{l-1}} \sum_{p=1}^P \sum_{q=1}^Q w_{k,c,p,q}^{(l)} a_c^{(l-1)} + b_k^{(l)}$$

where  $z_k^{(l)}$  is the output of the  $k$ -th feature map in the  $l$ -th layer,  $a_c^{(l-1)}$  is the activation of the  $c$ -th feature map in the  $(l-1)$ -th layer,  $w_{k,c,p,q}^{(l)}$  is the weight associated with the  $k$ -th feature map at position  $(p, q)$  in the  $l$ -th layer and the  $c$ -th feature map in the  $(l-1)$ -th layer,  $b_k^{(l)}$  is the bias term added to each neuron in the  $k$ -th feature map, and  $P$  and  $Q$  are the spatial dimensions of the filter. Figure 2.4 illustrates the convolution operation between an input (local receptive field) and a filter to produce an output. The filter slides over the local receptive field which is a small portion of the input image. At each step, the



element-wise multiplication between the filter and receptive field is computed, and the sum of these multiplications produces a single output element. The filter is then shifted to the right by a certain amount called the stride, and the process is repeated until the entire input has been convolved with the filter to produce the output which is the feature map.

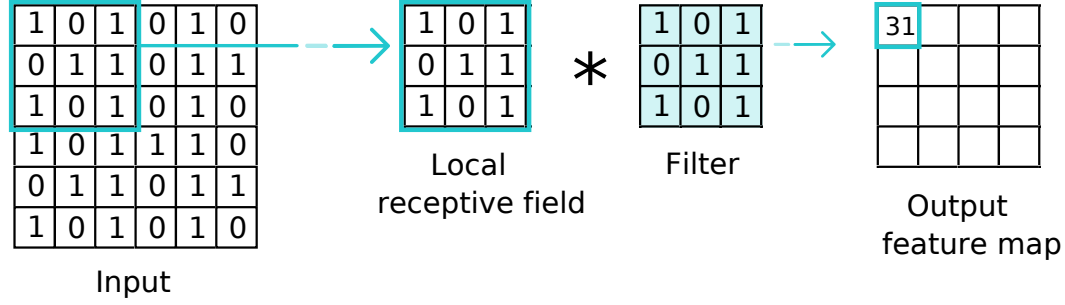


Figure 2.4: Convolution operation between input (local receptive field) and filter to produce feature maps.

As above-mentioned in Section 2.5.1 and in order to introduce some non-linear transformation to avoid possible linear transformation, the output of a convolutional layer should also be activated. ReLU ( $f(x) = \max(0, x)$ ) is the most used activation function in the context of CNNs.

- **Pooling layer:** This layer is *not parametrized*, it has no weight or bias to learn and optimize during the training process. Specifically, the pooling layer is used to simplify the network by connecting the input to a small square window of fixed width and height. The pooling layer also uses a *stride*, which defines the step size of the pooling window. By applying this, the pooling layer reduces the complexity of the network. Various methods have been used in the pooling layer, such as *Max-pooling*, *Min-pooling* and *Average-pooling* [33]. Figure 2.5 shows a simplified example of the three mentioned pooling functions. In our work, we used *Average-pooling* and *Max-pooling*, and they can be defined by:

$$\text{Avg-pooling}(x, S) = \frac{1}{hw} \sum_{i,j} (x_{i,j})$$

$$\text{Max-pooling}(x, S) = \max_{i,j} (x_{i,j})$$

where  $x$  is the input data (resulted feature maps),  $S$  is a region of size  $(h \times w)$  in  $x$ ,  $i$  and  $j$  are indices that iterate over the rows and columns of the region  $S$ , where  $\frac{1}{hw} \sum_{i,j}$  represents the average value of all the elements in the region  $S$ .  $\max_{i,j}$  represents the maximum value of all the elements in the region  $S$ .

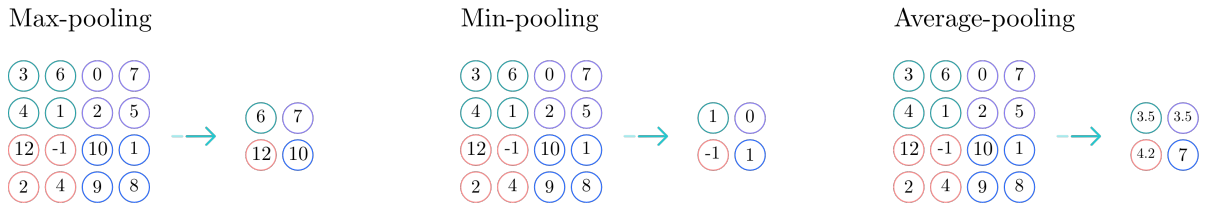


Figure 2.5: Example of pooling operations in CNNs. Max pooling selects the maximum value in each pooling window, while min pooling selects the minimum value and average pooling calculates the average value.

- Fully connected layer:** After the convolutional and pooling layers, the output is a set of feature maps that contain spatial information. In order to pass this information to a fully connected layer, the feature maps need to be flattened into a one-dimensional vector. This is done by stacking all the elements of each feature map on top of each other, resulting in a long vector. This flattened vector is then fed into a fully connected layer, which is a regular neural network layer where all the nodes are connected to the nodes in the previous layer. The mathematical formulation of a fully connected layer can be represented as:

$$z^{(l)} = w^{(l)} a^{(l-1)} + b^{(l)}$$

where  $w^{(l)}$  is the weight matrix of the layer,  $a^{(l-1)}$  is the output vector of the previous layer, and  $b^{(l)}$  is the bias vector. The output of the fully connected layer is then passed through an activation function before being fed into the next fully connected layer, or to produce an output prediction in the case of the last layer.

Fig. 2.6 illustrates the general flow of a CNN, which starts with the input data, followed by convolutional layers that extract feature maps, pooling layers that downsample the feature maps, flattening that reshapes the pooled feature maps into a vector, fully connected layers that apply a non-linear transformation to the vector, and finally, an output layer that produces the predictions

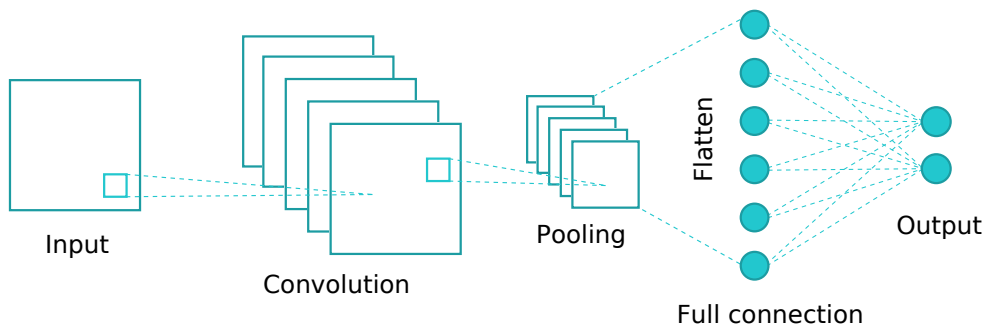


Figure 2.6: General flow of a CNN architecture

### 2.5.4 Long-Short Term Memory (LSTM)

In Section 2.5.1, we discussed RNNs, which have been introduced to overcome the limitation of ANNs in handling sequential data. RNNs have feedback connections that allow them to process sequences of variable length, and they can learn to model long-term dependencies in the input data. However, RNNs suffer from the issue of vanishing gradients when processing long sequences, which hinders their ability to capture long-term dependencies accurately [51]. To tackle this issue, specialized types of RNNs have been developed, such as LSTM and GRU architectures. One of the contributions in the present work is based on LSTM. In the following section, we will discuss LSTM architecture and how it overcomes the limitations of traditional RNNs and details the maths behind it [52]. RNNs struggle when the sequences become too long, where the gradients in the network become too small to update the weights effectively, this is called vanishing gradients, resulting in a degradation in performance. LSTM networks were introduced to address this limitation, by utilizing memory cells to maintain information over long sequences and gating mechanisms to control the flow of information through the cell. These mechanisms enable LSTMs to selectively forget or remember information, allowing them to model long-term dependencies more effectively than traditional RNNs. LSTM has three main components: *input gate*, *forget gate*, and *output gate*.

- **Input gate:** Responsible for controlling how much new information should be added to the cell state. It uses the current input and the previous hidden state as inputs and applies a sigmoid function to both to produce a value between 0 and 1 for each element of the cell state. An input gate value of 0 means that no new information is added to the cell state for that element, while a value of 1 means that all the new information is added. The input gate can also be thought of as a filter that controls which information from the current input should be allowed into the cell state, and which should be ignored. The input gate in the LSTM is defined by Equation 2.16. Equation 2.17 is used to compute the candidate cell state, and Equation 2.18 to compute the updated cell state.

$$(2.16) \quad i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$(2.17) \quad \tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$(2.18) \quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

In Equations 2.16, and 2.17,  $x_t$  presents the input vector at time step  $t$ ,  $h_{t-1}$  is the hidden state vector from the previous time step,  $\sigma$  presents the sigmoid activation function. The

weight matrix and bias term for the input gate are denoted by  $W_i$  and  $b_i$ , respectively. The input gate controls the amount of new input that is used to update the cell state by squashing the input between 0 and 1 using Equation 2.16. A gate value of 1 means that all of the input information passes through to the cell state, while a value of 0 means that all new input information are blocked. The input gate acts as a filter that determines which information from the current input should be added to the cell state and how much of it should be ignored. Equation 2.17 is used to compute the candidate cell state, starting by applying dot product operation to the concatenation  $[h_{t-1}, x_t]$  with  $W_c$ , then adding the bias term  $b_c$ , after that, applying TanH to squash the value between -1 and 1 to allow the candidate cell to capture a wide range of values. We denote that  $W_c$  and  $b_c$  are trainable parameters that are used to update the cell state. Lastly, Equation 2.18 is used to update the cell state using the computed  $\tilde{C}_t$ ,  $i_t$ ,  $C_{t-1}$  which is the cell state of the previous step, and  $f_t$  denoted by Equation 2.19 which is the forget gate activation,  $\odot$  denotes the element-wise multiplication.

- **Forget gate:** Determines which information to keep or discard from the cell state. It receives input from the previous hidden state  $h_{t-1}$  and the current input  $x_t$  and produces a forget vector  $f_t$  that ranges between 0 and 1 for each element of the cell state  $C_{t-1}$ . The forget gate uses  $\sigma$  that squashes the input to a value between 0 and 1 in order to determine how much of each cell state element to forget. The forget gate activation  $f_t$  is computed as:

$$(2.19) \quad f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where  $W_f$  and  $b_f$  represent the weight matrix and bias vector for the forget gate, respectively. The activation of the forget gate is used to compute the update of the cell state (Equation 2.18).

- **Output gate:** Controls the flow of information from the current hidden state to the output. The output gate takes as input the current input  $x_t$ , the previous hidden state  $h_{t-1}$ , and the current cell state  $C_t$ , and produces the output vector  $h_t$  which is the hidden state of timestep  $t$ . It is responsible for producing the output of the LSTM cell. Mathematically, the output gate  $o_t$  is defined by Equation 2.20:

$$(2.20) \quad o_t = \sigma(W_o \cdot [h_{t-1}, x_t, C_t] + b_o)$$

where  $W_o$  and  $b_o$  are the weight matrix and bias vector of the output gate. The output vector  $h_t$  is computed using Equation 2.21:

$$(2.21) \quad h_t = o_t \odot \tanh(C_t)$$

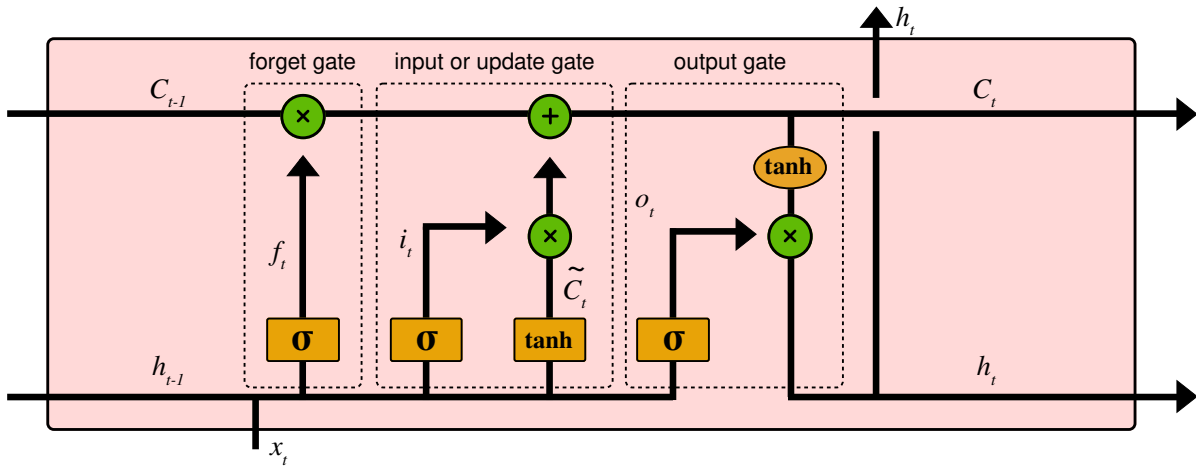


Figure 2.7: General structure of LSTM cell

### 2.5.5 Attention Mechanism

The attention mechanism has been widely used in various DL applications, including NLP, CV, and SR. Similar to how human attention works, this mechanism allows the model to pay more focus to certain important parts of the input while ignoring others. The attention mechanism can be applied by assigning a weight to each input based on its significance to the treated task. The fact of using attention mechanism in DL models has significantly improved the performance of these models by adding the ability to extract and capture more complex relationships between input and output. This concept has been widely used in different applications such as Natural Language Processing (NLP) [2, 14, 15, 139], Speech Recognition (SR) [20, 24, 73, 141], and Computer Vision (CV) [53, 144, 148, 149]. Regular neural networks suffered from fixed-length encoding vectors of data, where input data may have an equal importance degree, which can lead to the model's performance loss [122]. On the other hand, the key to the attention mechanism concept is to assign different importance (weights) to input data, which allows the model to learn from relevant and important information and enhance its performance. Several benefits from using the attention mechanism with DL models, such as faster convergence, by focusing on the most important parts of the input data, the model can show quick convergence, and more interpretability, where different applications need to interpret their predictions such as finance, and by focusing on the most relevant parts of the data, the prediction results can be more interpretable. There are several types of attention mechanisms, including *additive attention*, and *scaled dot-product attention*. In the rest of this section, we will explore *additive attention*, which is the attention type that has been used in our work.

Additive attention was introduced in *Bahdanu Attention* [11] which is used for sequence-to-sequence for machine translation tasks. It is based on three phases: *score alignment*, *weight computation*, and *context vector computation*, or what we call later *attention vector*. It uses a

feedforward neural network to compute the attention scores. However, in our work, we used a simplified variant of additive attention where the attention weights are computed by taking the dot product of the input with a weight matrix, followed by applying a bias term and a hyperbolic tangent activation function. The alignment scores are then obtained by taking the dot product of the attention weights with the input. Given the hidden state of the LSTM layer  $H_t = [h_1, h_2, \dots, h_t]$ ,  $h_t$  presents the hidden state of the encoder at timestep  $t$ , and the input of the attention layer. Scores alignment phase is represented by learnable weights  $W_{att}$  and  $b_{att}$ . Scores alignment is defined by Equation 2.22.

$$(2.22) \quad S_t = \tanh(W_{att}h_t + b_{att})$$

After computing the scores alignment  $S_t$  which presents how well each element in the input sequence aligns with the current hidden state  $h_t$ , Softmax function is used to compute *weights*  $\alpha_t$  to transform and normalize the alignment scores into a probability distribution, using Equation 2.23.

$$(2.23) \quad \alpha_t = \frac{\exp(S_t)}{\sum_{j=1}^T \exp(S_j)}$$

where  $T$  is the total length of the input sequence. The attention weights determine how much focus to place on each element of the input sequence at time step  $t$ .

After computing attention weights  $\alpha_t$ , we then compute the context vector, denoted as the *attention vector*  $a_t$ , defined by Equation 2.24.

$$(2.24) \quad a_t = \sum_{j=1}^T \alpha_{t,j}h_j$$

## 2.6 Overfitting and Normalization

One common issue in ML is *overfitting*, which occurs when the model becomes too complex and starts to memorize the training data instead of learning generalizable patterns. Preventing overfitting is crucial in ML and DL, to ensure that the model generalizes well to unseen and new data. Normalization techniques, such as batch normalization, can also be used to prevent overfitting by scaling the inputs to have zero mean and unit variance, which can improve the convergence of the training process.

### 2.6.1 Dropout

Dropout is a regularization technique that randomly drops out some nodes in the neural network during training, which helps prevent overfitting. During training, at each update of the network

weights, a neuron is either dropped out with probability  $p$  or kept with probability  $1 - p$ . If a neuron is dropped out, its output is multiplied by 0, effectively setting it to zero, and the signal is not propagated through it during that particular forward or backward pass. This introduces random drop network weight, effectively creating a different and more robust model at each update and preventing overfitting.

### 2.6.2 Batch Normalization (BN)

BN is applied to the inputs of each layer in a mini-batch during the training phase [56]. It works by normalizing the mean and variance of the inputs to a neural network layer, thereby reducing the effect of the scale of the weights and the input data. Additionally, it helps in preventing overfitting, as it reduces the internal covariate shift, which is the phenomenon of the distribution of each layer's inputs changing during training. BN layer output is given by Equations 2.25 and 2.26.

$$(2.25) \quad y^k = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}$$

$$(2.26) \quad \hat{x}^k = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

where  $x^{(k)}$  is the input to the  $k$ -th layer,  $\mu_B$  and  $\sigma_B$  are the mean and standard deviation of the input over the mini-batch  $B$ , and  $\epsilon$  is a small constant added to avoid division by zero. The normalized input  $\hat{x}^{(k)}$  is then scaled and shifted using learned parameters  $\gamma^{(k)}$  and  $\beta^{(k)}$ .

## 2.7 Overview on Audio Features

Audio refers to the sound waves that are perceived by the human ear. These sound waves can be represented as an electrical signal known as an audio signal. An audio wave is a representation of how a sound wave varies over time and can be graphed as an amplitude vs. time plot, as exemplified in Figure 2.8, while audio features present the mathematical representations of these audio signals that can be used to characterize various aspects of the sound such as pitch, loudness, timbre, and extract more useful information which can be used for several tasks such as classification, synthesis, clustering, segmentation, and more. In our work, we focus on the extraction of audio features from cough sound, which are used as input to our ML and DL models to build COVID-19 diagnosis systems based on solely cough sound. The question may arise as to why we should focus on extracting audio features when DL models can automatically extract features and classify audio data. While it is true that DL models have shown impressive results in audio classification tasks, utilizing raw audio signals without any feature extraction may lead to the suboptimal performance of the model. Raw audio signals contain a large amount of redundant and irrelevant information, which can make it difficult for ML or DL models to extract

the most important features for a given task. Additionally, raw audio signals may suffer from issues such as noise, variability in recording conditions, and speaker characteristics, which can further complicate the feature extraction process. Therefore, extracting relevant audio features can provide a more informative representation of the audio data, leading to better results for the given task.

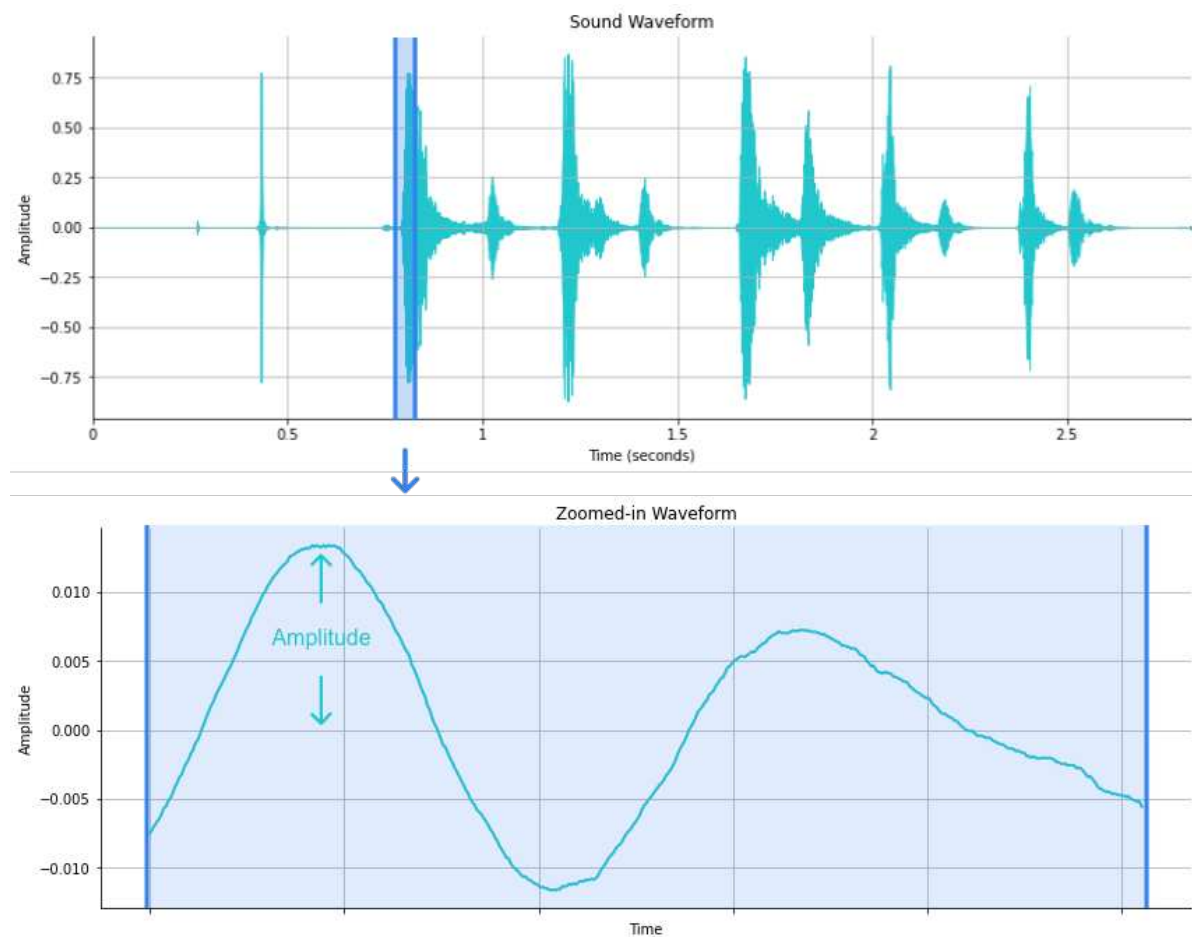


Figure 2.8: Illustration of a sound wave showing the variation of amplitude over time. The zoomed-in section highlights the detailed variations in signal amplitude.

There are several types of audio features that can be used depending on the task to be performed, including *time-domain* and *frequency-domain*. These categories are mostly used in our work, thus, in the rest of this section, a comprehensive study will be conducted on various audio features that belong to the two mentioned types of features.

### 2.7.1 Time domain features

Can be used to characterize the properties of an audio signal. These features are computed directly from the raw audio waveform and include various statistics such as the amplitude,



energy, and temporal characteristics of the signal.

### 2.7.1.1 Zero Crossing Rate (ZCR)

ZCR indicates the number of times the audio signal crosses the zero amplitude level. It is a simple and effective feature for detecting the presence of high-frequency components in a signal. ZCR is defined by equation Equation 2.27 [38, 41].

$$(2.27) \quad ZCR = \frac{1}{N-1} \sum_{t=1}^{N-1} |sgn(x(t)) - sgn(x(t-1))|$$

where  $N$  is the number of samples in the audio signal,  $x(t)$  represents the audio signal at time  $t$ , and  $sgn(\cdot)$  is the signum function, which returns the sign of a number as +1, 0, or -1 for positive, zero, or negative numbers respectively.

### 2.7.1.2 Root Mean Square Energy (RMSE)

Measures the root mean square of the signal amplitude over time. It provides a measure of the overall energy level of the signal [38]. RMSE is defined by Equation 2.28.

$$(2.28) \quad RMSE = \sqrt{\frac{1}{N} \sum_{t=1}^N x^2(t)}$$

where  $x(t)$  is the signal amplitude at time  $t$  and  $N$  is the total number of samples in the signal. The resulting value is expressed in the same units as the original signal amplitude, such as dB or volts.

### 2.7.1.3 Temporal Centroid

Indicates the center of gravity of the signal's energy distribution over time [66]. It is calculated by weighting each time sample with its energy and computing the mean of the weighted time values. Temporal Centroid is defined by Equation 2.29.

$$(2.29) \quad TemporalCentroid = \frac{\sum_{t=1}^N t|x(t)|^2}{\sum_{t=1}^N |x(t)|^2}$$

where  $N$  is the number of samples in the signal and  $x(t)$  is the amplitude of the signal at time index  $t$ .

## 2.7.2 Frequency domain features

These are other types of features that describe the distribution of energy of a signal in the frequency domain. Computing these features is based on an essential step which is transforming the signal from the time domain into the frequency domain using techniques such as Fourier

Transform (FT), or Short-Time Fourier Transform (STFT). This type of feature can provide information about the *spectral* characteristics of the audio signal, which refers to the distribution of energy or power across different frequencies present in the signal. It provides information about how much of the signal's energy is concentrated at different frequencies. In the following, we will detail the mostly used frequency-domain features in our contributions.

### 2.7.2.1 Mel-spectrogram

The raw audio signal captures the amplitude of the sound wave over time, however, it cannot provide information about the individual frequencies present in the signal. To extract the frequency information, Fast Fourier Transform (FFT) can be used, which decomposes the signal into its individual frequencies and the frequency's amplitude which results in the *Spectrum*, as illustrated in Figure 2.9. However, in most of the audio signal cases such as speech and music, signals are non-periodic, which vary over time, in other words, the spectrum obtained by applying the FFT on the whole signal provides frequency information for the entire signal duration, and not any time-varying changes in frequency content, thus, the need to represent the spectrum of the signal as it varies over time is raised.

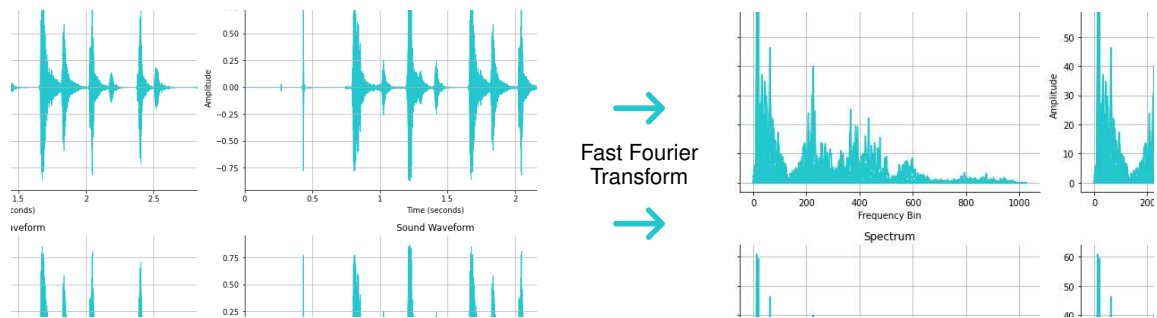


Figure 2.9: Illustration of a sound wave and the spectrum which is the result of applying FFT.

With this respect, STFT can be applied to present the *spectrum over time*, which consists of applying FFT of several windowed segments of the signal. The windowing function such as *Hanning*, or *Han*, is applied to the signal to reduce spectral leakage which is caused by the abrupt start and end points of the signal. The windowing function smooths out these points by gradually increasing and decreasing the amplitude of the signal within the window. The STFT is performed with a sliding window over the signal, and the result is called a *Spectrogram*, which is a visual representation of the signal's amplitude as it varies over time at different frequencies. STFT of an input signal  $x(n)$  can be defined by Equation 2.30 [59].

$$(2.30) \quad X(m, k) = \sum_{n=0}^{N-1} x(n)w(n-m)e^{-i2\pi kn/N}$$

where  $w(n-m)$  is the window function centered at  $m$ ,  $N$  is the length of the FFT which refers to the window size,  $k$  is the frequency bin index which usually ranges from 0 to  $(N/2) + 1$ ,  $i$  is the

imaginary unit (i.e.  $\sqrt{-1}$ ), while  $X(m, k)$  presents the frequency representation of the signal at time  $m$  and frequency bin  $k$ .

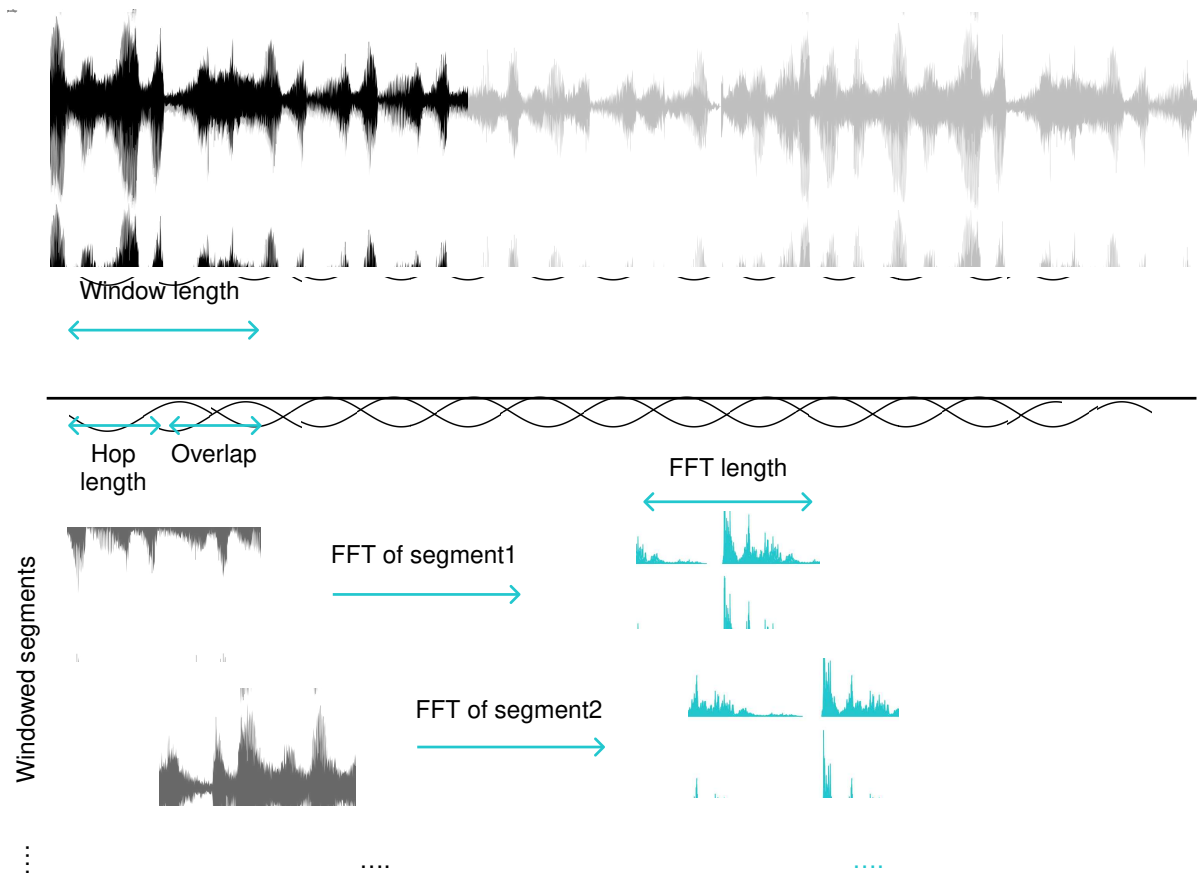


Figure 2.10: Illustration of the STFT applied on an input signal. The input signal is divided into overlapping windows of fixed length (window size) and shifted by a fixed interval (hop length) to produce a sequence of windowed segments. Each segment is windowed using a window function and then, spectrum is computed using FFT with a chosen FFT length. The amount of overlap between adjacent windows (overlap length) affects the time-frequency resolution trade-off in the resulting *spectrogram*.

Furthermore, Figure 2.10 presents a summary of the main steps to produce a *Spectrogram*, which can be represented in 2-dimensional visualization by stacking the resulting spectrums on top of each other, where the x-axis presents time and the y-axis presents frequency as shown in Figure 2.11.

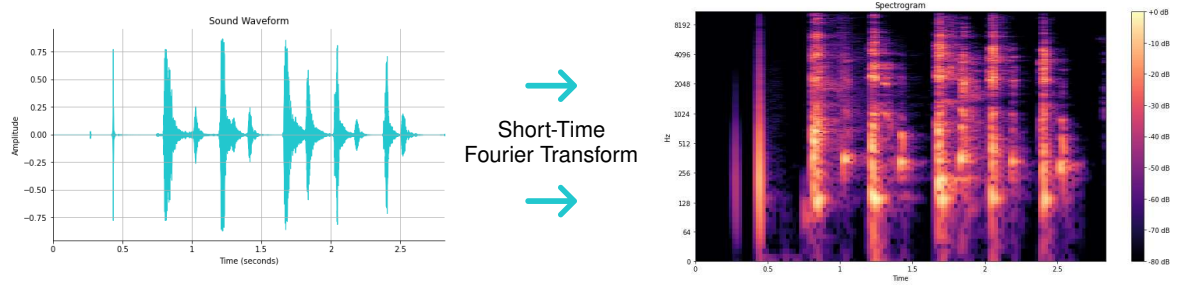


Figure 2.11: Example of an input signal wave and the resulting *Spectrogram*, where the y-axis is converted to log-scale and the color dimension to decibels.

Moreover, studies have shown that humans do not perceive frequencies on a linear scale, thus, the Mel-scale [121] is put forward, which is a unit of pitch that is based on how humans perceive sound. *Mel-spectrogram* is a spectrogram where the frequencies are converted to the Mel-scale using Equation 2.31.

$$(2.31) \quad f_{mel} = 2595 \log_{10} \left( 1 + \frac{f}{700} \right)$$

where  $f$  is the frequency in Hz and  $f_{mel}$  is the frequency in Mel-scale. Figure 2.12 shows the application of Mel-scale on an input spectrogram.

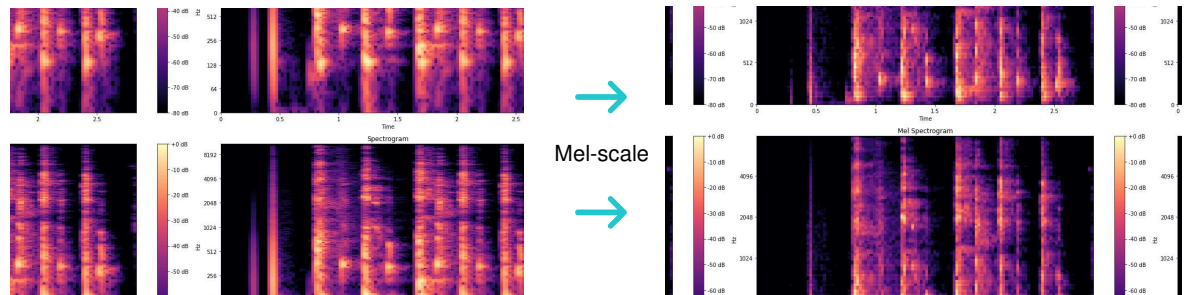


Figure 2.12: Example of an input spectrogram and the resulting *Mel-spectrogram*.

### 2.7.2.2 Spectral Centroid

Similarly to Temporal Centroid, Spectral Centroid measures the center of gravity of the spectrogram and indicates the frequency at which the energy of a signal is concentrated. Spectral Centroid can be mathematically defined by Equation 2.32.

$$(2.32) \quad SC = \frac{\sum_{i=1}^n f(i) \times |X(i)|}{\sum_{i=1}^n |X(i)|}$$

where  $X(i)$  is the magnitude value in the  $i$ -th frequency bin,  $f(i)$  is the frequency corresponding to the  $i$ -th frequency bin in the spectrogram, and  $n$  is the number of frequency bins.

### 2.7.2.3 Spectral roll-off

Spectral Roll-off measures the frequency below which a certain percentage of the total spectral energy lies, which can be calculated with  $SR = f_{roll-off}$ , where  $f_{roll-off}$  is the frequency at which the cumulative spectral energy,  $E_{cum}(f)$ , reaches the desired percentage, and  $f$  is the frequency. The cumulative spectral energy is defined by Equation 2.33.

$$(2.33) \quad E_{cum}(f) = \sum_{i=1}^n |X(i)|^2$$

where  $X(i)$  is the magnitude value in the  $i$ -th frequency bin, and  $n$  is the number of frequency bins in the spectrogram.

### 2.7.2.4 Mel-Frequency Cepstral Coefficients (MFCCs)

MFCCs are a set of features commonly used in speech and audio processing, derived from the spectral envelope of a sound signal. They capture the most relevant spectral characteristics for the perception of speech and music. MFCCs are computed through several steps, and can be summarized by Equation 2.34.

$$(2.34) \quad MFCCs = DCT(\log_{10}(MelSpectrum))$$

where *MelSpectrum* is the Mel-frequency spectrum of the signal, and DCT is the Discrete Cosine Transform. The logarithm is taken to compress the dynamic range of the spectrum and to emphasize the spectral envelope.

## 2.8 Dimensionality Reduction

Dimensionality Reduction is a data analysis technique often used in ML, which consists of reducing the number of features or variables in a dataset, where data is involved in a process of transformation, from higher-dimensional space into lower-dimensional space by preserving the most of variance. It aims to eliminate irrelevance, and redundancy and have a more comprehensive way to visualize data. As discussed in Section 2.2.1, dimensionality reduction can be used in unsupervised learning tasks, additionally, it has been reported to be useful for supervised tasks [58, 75]. Various methods have been developed and used in different applications of dimensionality reduction including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), t-distributed Stochastic Neighbourhood Embedding (t-SNE), LLE, and Autoencoders. This section will give a comprehensive introduction to the methods that have been widely used in this thesis, LLE, and Autoencoders.

### 2.8.1 Locally Linear Embedding (LLE)

Locally Linear Embedding (LLE) consists of a nonlinear dimensionality reduction technique that seeks to preserve the local structure of data [101]. Finding a lower-dimensional representation of

the data that preserves the local relationships between neighboring points is the idea behind LLE, where weight is defined for each data point, and the weights are chosen to minimize the squared difference between each data point and its reconstruction. Mathematically, it can be defined by Equation 2.35.

$$(2.35) \quad \min_W \sum_{i=1}^N |x_i - \sum_{j=1}^N W_{ij}x_j|^2 \quad \text{s.t.} \quad \sum_{j=1}^N W_{ij} = 1$$

where  $W$  is the weight matrix, which maps each point  $x_i$  to a set of its neighbors. The weight  $W_{ij}$  reflects the degree to which point  $x_j$  contributes to the reconstruction of point  $x_i$ . The constraint  $\sum_{j=1}^N W_{ij} = 1$  ensures the reconstruction of each point is a linear combination of its neighbors.

After that, the algorithm tries to find a lower-dimensional projection  $Y$  of the input data  $X$  that preserves the weight matrix  $W$ . Equation 2.36 defines an optimization problem where  $y_i$  is the lower-dimensional projection of  $x_i$ , and the aim is to find a new set of data points  $y_1, y_2, \dots, y_N$  that preserve the local geometry of the original points  $x_1, x_2, \dots, x_N$ , as measured by  $W$ .

$$(2.36) \quad \min_Y \sum_{i=1}^N |y_i - \sum_{j=1}^N W_{ij}y_j|^2 \quad \text{s.t.} \quad \begin{cases} \sum_{i=1}^N y_i = 0 \\ \sum_{i=1}^N y_i y_i^T = I \end{cases}$$

where  $y_i$  is the embedding of the data point  $x_i$  and  $I$  is the identity matrix. The first constraint ensures that the embedding is centered at the origin, while the second constraint ensures that the embedding has unit covariance. Figure 2.13 illustrates the three main steps of LLE. In the first step, nearest neighbors are selected for each data point. In the second step, the data is reconstructed using linear weights. In the third step, the reconstructed data is mapped to an embedded coordinate system using an optimization algorithm. The resulting embedded coordinates represent a low-dimensional representation of the original high-dimensional data.

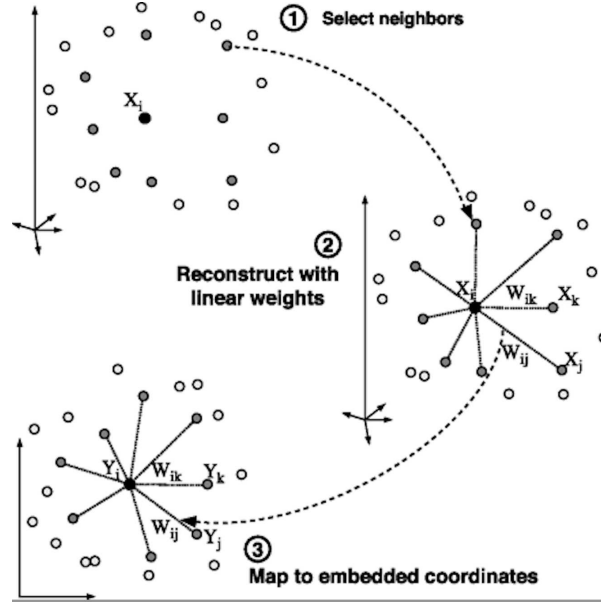


Figure 2.13: Illustration of the three main steps of LLE [101].

### 2.8.2 Autoencoders

Autoencoder consists of an ANN that is widely used for unsupervised learning tasks, with a particular focus on dimensionality reduction. In simple words, an autoencoder is a neural network that learns to compress and then reconstruct different types of input data, while minimizing the reconstruction error. Autoencoders can be used to extract meaningful features and patterns from high-dimensional data, which is often useful for dimensionality reduction, data compression, and visualization. An autoencoder consists of three parts: an *encoder*, *decoder*, and *latent space*. The encoder network takes input data and maps it to a compressed representation, also called latent space, or code. Encoder network can be presented mathematically by Equation 2.37 [39].

$$(2.37) \quad z = f_{\theta_{encoder}}(x)$$

where  $z$  is the latent space representation,  $x$  is the input data,  $f_{\theta_{encoder}}$  is the encoder function which involves learnable weights  $\theta_{enc}$ .

Decoder network takes the compressed representation  $z$  and maps it back to the original input data. Decoder network can be defined by Equation 2.38.

$$(2.38) \quad \hat{x} = g_{\theta_{decoder}}(z)$$

where  $\hat{x}$  is the reconstructed data,  $g_{\theta_{decoder}}$  is the decoder function with learnable weights  $\theta_{decoder}$ .  $g$  is denoted as a non-linear transformation that aims to reconstruct the original input with

minimal error from the latent representation  $z$ . Thus, this is commonly achieved by minimizing a reconstruction loss between the input  $x$  and the reconstructed output  $\hat{x}$ . Various loss functions have been used depending on the data and the task, including MSE, BCE, and Huber loss. A general formulation of the loss function in an autoencoder is given by Equation 2.39.

$$(2.39) \quad Loss(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^n L(x_i, \hat{x}_i)$$

where  $x$  is the input data,  $\hat{x}$  is the reconstructed data,  $n$  is the number of samples, and  $L$  is the used loss function.

In the training process, an optimization algorithm, such as Stochastic Gradient Descent (SGD), is used to minimize  $Loss(x, \hat{x})$ . There are various variations of autoencoders that have been developed to address specific tasks. One such variation is *deep autoencoder*, which is similar to a DNN, where it includes multiple hidden layers, allowing it to learn more complex representations of the input data. Figure 2.14, shows a general structure of a deep autoencoder with two hidden layers.

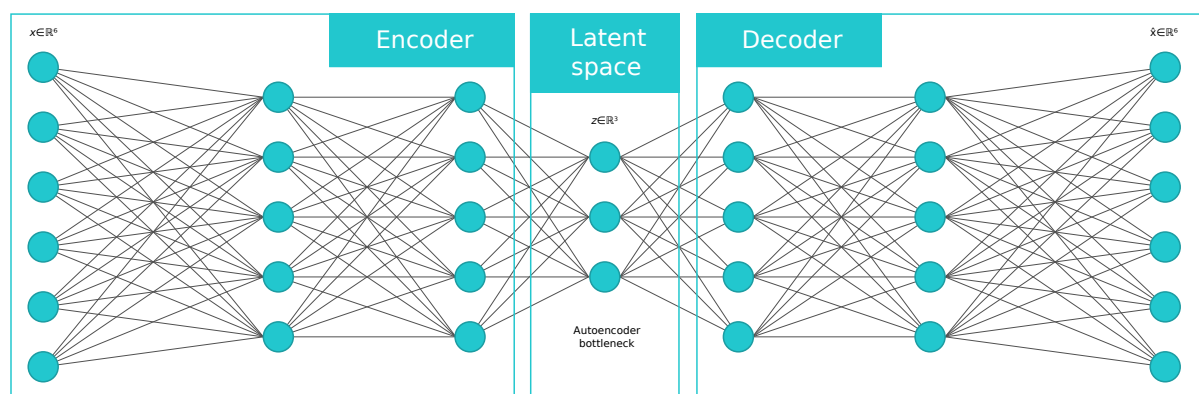


Figure 2.14: General structure of a deep autoencoder with two hidden layers, a latent space of size 3, and an input size of 6. The encoder maps the input to a compressed representation in the latent space, while the decoder maps the latent space back to the original input space.

## 2.9 Data Augmentation Methods

Data augmentation reflects the synthetic increase of dataset size, by generating new samples based on the existing ones. It is widely used as it has shown promising improvement in many ML models [1, 10, 30, 82]. This technique is used to overcome the lack of data, dataset skew, or imbalanced datasets, where the number of samples in one class is much smaller than in the other class. Data augmentation can be applied to different types of data such as tabular data, textual data, images, and audio. By applying this technique, ML models can learn from a larger and more diverse set of data, leading to better performance and generalization on new and unseen data. In our contributions, data augmentation was an essential step to enhance our ML and DL



models, thus, SMOTE for tabular data and audio-specific data augmentation techniques will be explored and discussed in this section.

### 2.9.1 SMOTE

SMOTE [23] is an algorithm that mainly addresses the issue of imbalanced datasets. However, it can also be a data augmentation algorithm, it is suitable for tabular data and it works by generating synthetic samples of the minority class to balance the dataset, thus improving the performance of ML models. The idea behind SMOTE is based on selecting an instance from the minority class, finding its  $k$  nearest neighbors in the same class then, creating a synthetic instance by choosing one random neighbor from the  $k$  neighbors and then, by connecting the two instances, a line segment is created in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances, where the combination is controlled by a random number between 0 and 1. This process is repeated until the desired level of balance is achieved. Algorithm 5 formally describes the SMOTE process.

---

#### Algorithm 5 SMOTE Algorithm [23]

---

**Require:**  $X$ : The minority class samples,  $N$  number of synthetic samples to be generated, and the number of nearest neighbors to consider  $k$

**Ensure:**  $N$  New synthetic minority class samples

- 1: Create an empty dataset for the synthetic samples  $S$ ;
- 2: **for**  $i \leftarrow 1$  to  $N$  **do**
- 3:   Randomly choose a minority class sample  $x_i$ ;
- 4:   Find  $k$ -nearest neighbors for  $x_i$  from the minority class;
- 5:   Randomly choose one of the  $k$ -nearest neighbors  $x_{nn}$ ;
- 6:   Generate new synthetic sample  $x_{new}$  as a convex combination of  $x_i$  and  $x_{nn}$ :

$$x_{new} = x_i + \epsilon(x_{nn} - x_i)$$

where  $\epsilon$  is a random number between 0 and 1;

- 7:   Add  $x_{new}$  to the synthetic dataset  $S$ ;
  - 8: **end for**
  - 9: **return** Synthetic minority class samples  $S$ ;
- 

### 2.9.2 Audio data Augmentation

Audio data augmentation is the creation of new variations of existing audio data by applying different transformations to the original signal to enhance the performance of the trained ML and DL models. By applying various transformations to the original audio signals, the resulting augmented signals can capture different variations of the same underlying signal, making the models more robust to different variations and distortions in the input data. Audio data augmentation techniques can be categorized into two main types: *signal-based augmentation*, and

*spectral-based augmentation*. In the first category, *Pitch-shifting*, *Time-shifting*, *Time-stretching*, and *White-noise*, and *SpecAugment* for the second category will be discussed in the rest of this section.

- Pitch-shifting:** Consists of changing the pitch, or the frequency, without changing the audio duration, where the frequency of each spectral component in the audio signal is scaled by a factor to raise or lower the pitch, and this can be achieved by resampling the signal at a higher or lower sampling rate. Let  $x[n]$  be the input signal, and  $y[n]$  be the output signal after pitch-shifting by a factor  $\alpha$ , which represents a semi-tone by unit. The relation between  $x[n]$  and  $y[n]$  can be expressed by  $y[n] = x[n/\alpha]$ , where  $n$  is the sample index and  $\alpha$  is the shifting factor. If  $\alpha$  is greater than 1, the pitch is shifted upwards, and if  $\alpha$  is less than 1, the pitch is shifted downwards. Figure 2.15 shows an example of original and pitch-shifted signals.

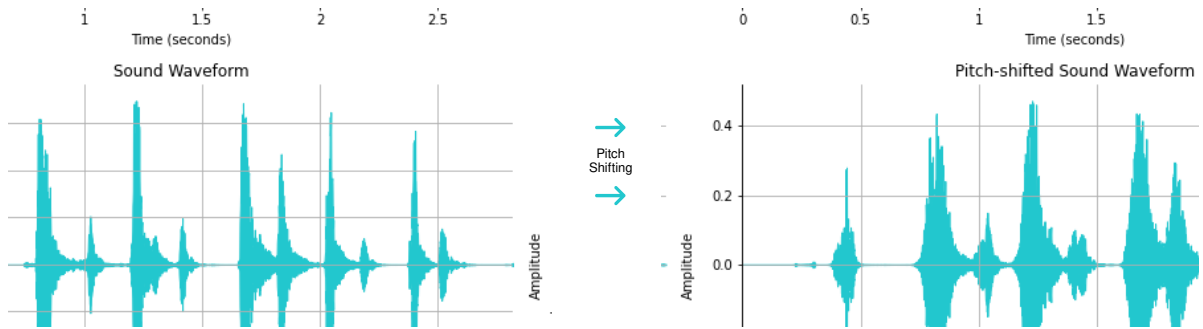


Figure 2.15: Pitch-shifting transformation with  $\alpha = 3$ .

- Time-shifting:** Involves shifting the signal along the time axis. This can be achieved by adding or removing some silence or audio from the beginning or the end of the signal. The main purpose of time-shifting is to introduce temporal variability in the data. Given an audio signal  $x[n]$ , a time-shifted version of the signal  $x$  can be represented as  $x'(n) = x[n - n_0]$ , where  $n_0$  is the amount of time to shift the signal by. Figure 2.16 shows an example of original and time-shifted signals.



Figure 2.16: Time-shifting transformation with  $n_0 = 0.5$ .

- **Time-stretching:** Consists of changing the duration of a signal without affecting the pitch. It can be represented by changing the signal sampling rate by a certain factor  $s$ . It can be formulated by  $x'[n] = x[m]$  where  $m = \text{round}(n * s)$ . The value of  $s$  determines the amount of time stretching or compression applied to the signal. If  $s < 1$ , the signal is stretched, otherwise, the signal is compressed. Figure 2.17 shows an example of original and time-stretched signals.

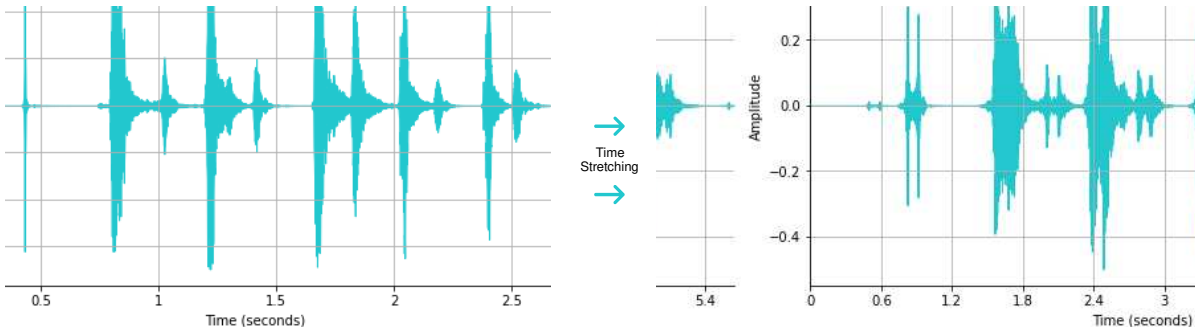


Figure 2.17: Time-stretching transformation with  $s = 0.5$ .

- **White noise:** Adds random noise to the signal. This can help to make the model more robust to noise and other possible distortions. Background white noise has a constant power spectral density, which means that it has an equal amount of energy across all frequencies. This can be achieved by adding a random sequence of values that are sampled from a normal distribution. The added noise can be represented by  $noise[n]$ . The noised signal  $y[n]$  is computed by  $y[n] = x[n] + noise[n]$ . Figure 2.18 shows an example of original and noised signals.

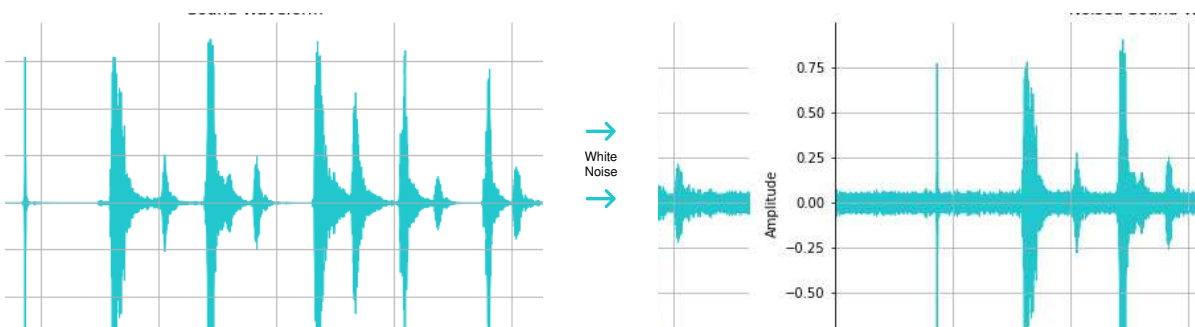


Figure 2.18: White noise addition with a standard deviation  $\sigma = 0.02$ .

- **SpecAugment:** Belongs to spectral augmentation category, introduced by Google Brain research [95]. SpecAugment is designed to randomly mask time-frequency spectrogram segments of the audio signal during training. This process helps the model learn to recognize informative patterns, even when some parts of the signal are missing. Consequently, the

model is more likely to prevent overfitting and improve generalization. SpecAugment technique uses three steps: *time warping*, *time masking*, and *frequency masking*.

Time warping involves shifting a random sequence of consecutive time steps in the spectrogram to the left or right, causing the spectrogram to warp in time. This technique is designed to improve the model's robustness to temporal shifts. The extent of the warping is controlled by a time warp parameter  $W$ . Time warping is described by Equation 2.40.

$$(2.40) \quad S_w(f, t) = S(f, g(t))$$

where  $S$  is the given spectrogram,  $S_w$  is the time-warped spectrogram,  $f$  is the frequency index,  $t$  is the time index, and  $g(t)$  is a function that maps the original time index  $t$  to the warped time index.

Time masking involves masking a random sequence of consecutive time steps in the spectrogram, while frequency masking involves masking a random sequence of consecutive frequency bins in the spectrogram. Both types of masking can be applied independently, and they are based on  $T$  and  $F$ , which are parameters of *Time masking*, and *Frequency masking*, respectively.  $t_r$  and  $f_r$  present the number of timesteps and mel frequency channels that will be masked, and they are chosen from 0 to  $T$ , and 0 to  $F$ , respectively. SpecAugment can be formulated by, first, *time masking* by Equation 2.41.

$$(2.41) \quad S_t(f, t) = \begin{cases} S(f, t) & \text{if } t < t_0 \text{ or } t \geq t_0 + t_r \\ 0 & \text{otherwise} \end{cases}$$

where  $S$  is the given spectrogram,  $S_t$  is the time-masked spectrogram,  $f$  is the frequency index,  $t$  is the time index, and  $t_0$  is chosen from  $[0, \tau - t_r)$  where  $\tau$  presents the number of timesteps, and second, *frequency masking* which can be defined by Equation 2.42.

$$(2.42) \quad S_f(f, t) = \begin{cases} S(f, t) & \text{if } f < f_0 \text{ or } f \geq f_0 + f_r \\ 0 & \text{otherwise} \end{cases}$$

where  $S_f$  presents the frequency-masked version of the spectrogram  $S$ ,  $f_0$  is chosen from  $[0, \nu - f_r)$ , where  $\nu$  presents the number of mel frequency channels. By using time and frequency masking to the spectrogram, SpecAugment technique can effectively increase the variability of the training dataset, and also, tackle the class imbalance issue. Figure 2.19 exemplified the result of applying time and frequency masking on an input spectrogram.

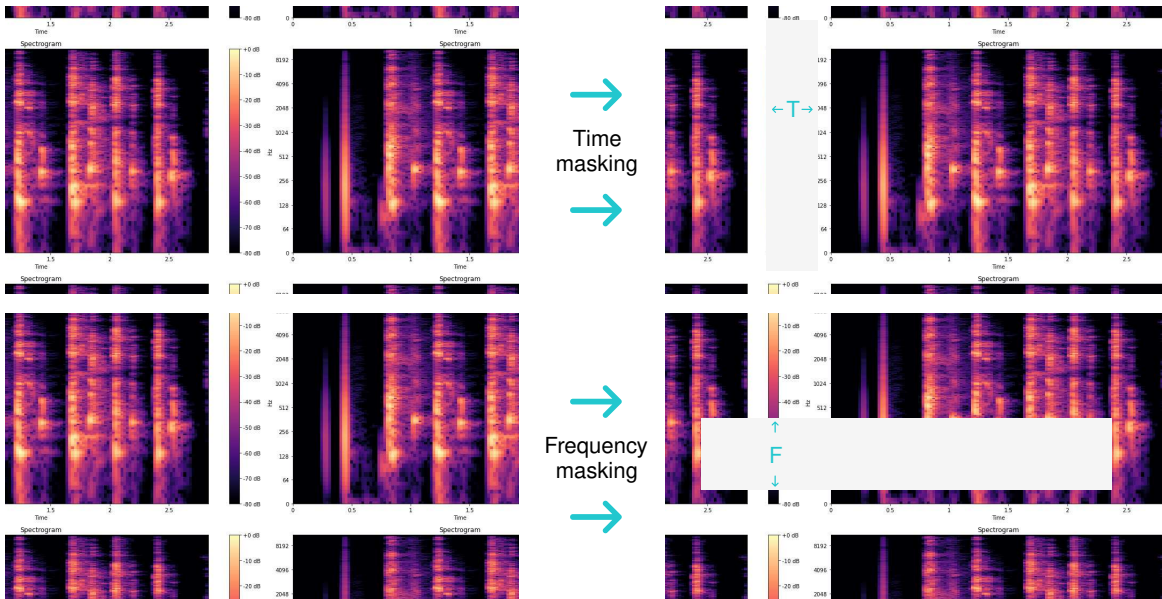


Figure 2.19: Simple illustration of SpecAugment technique.

## 2.10 ML and DL for respiratory diseases diagnosis

In this section, different recent works that have used ML and DL for different respiratory diseases diagnosis, excluding COVID-19, will be explored. We aim is to demonstrate the effectiveness of these techniques in accurately diagnosing respiratory diseases, which can greatly motivate our findings.

A. Manickam et al. [78] proposes a novel deep learning approach for the automatic detection of pneumonia using deep transfer learning to simplify the detection process with improved accuracy. The study preprocesses input chest X-ray images to identify the presence of pneumonia using U-Net architecture-based segmentation and classifies pneumonia as normal and abnormal (Bacteria, viral) using pre-trained on ImageNet dataset models such as ResNet50, InceptionV3, InceptionResNetV2. The dataset used consists of over 5300 images. The results showed that the proposed ResNet50 model achieved 93.06% of accuracy, 88.97% of precision, 96.78% o sensitivity, and an F1-score of 92.71%, which is higher than other models compared.

In the context of Chronic Obstructive Pulmonary Disease (COPD), S. Swaminathan et al. [125] developed a machine learning algorithm for the early detection of exacerbations in patients with COPD. The algorithm outperformed individual pulmonologists in identifying exacerbations and predicting appropriate triage action. The algorithm used simulated data for outpatient triage and evaluation of patients with COPD. The top two performing algorithms were the Gradient-Boosted Decision Tree and the Logistic Regression. The algorithm agreed with the consensus opinion in 88% of triage cases and had a success rate of 97% in determining if an exacerbation had occurred. The algorithm can improve the accuracy and consistency of COPD

triage and exacerbation prediction, and provide simple, easily accessible, safe, and highly accurate at-home decision support for patients. The study used 2501 patient scenarios for training and 101 cases for validation. E. Showkatian et al. [115] developed and compared the performance of a CNN model trained from scratch and a transfer learning-based approach to automatically detect Tuberculosis (TB) from chest X-ray images. Two publicly available datasets from Maryland and China with a total of 800 images, were used to train and evaluate the proposed models. The performance of the models was evaluated using five performance metrics. The proposed CNN architecture achieved an accuracy of 87%, while transfer learning-based models, including InceptionV3, Xception, ResNet50, VGG19, and VGG16, achieved higher accuracy, precision, sensitivity, F1-score, and ROC-AUC. Xception, ResNet50, and VGG16 models showed the highest performance for automated TB classification. The study provides a transfer learning approach with deep CNNs that can effectively classify TB and normal cases from chest radiographs.

In another context, A. G. Taylor et al. [127] aimed to create a large annotated dataset of chest X-rays containing pneumothorax and to train deep CNN to prioritize X-rays with potentially emergent moderate or large pneumothorax. A total of 13,292 frontal chest X-rays, including 3,107 with pneumothorax, were annotated by radiologists to train and evaluate multiple network architectures. Using an internal validation set, the two top-performing models were selected and evaluated on a held-out internal test set and an external National Institutes of Health ChestX-ray14 set. The "high sensitivity model" produced a sensitivity of 0.84, specificity of 0.90, and AUC of 0.94, while the "high specificity model" showed a sensitivity of 0.80, specificity of 0.97, and AUC of 0.96.

Regarding lung cancer, several studies have been conducted, Q. Song et al. [120] introduce three deep neural networks (CNN, DNN, and Stacked AutoEncoder (SAE)) for classifying lung cancer calcifications in Computed Tomography (CT) images. The LIDC-IDRI database, which includes 244,527 images from 1010 cases, was used to train and evaluate the performance of these networks. The networks were adjusted to classify benign and malignant lung nodules, and the CNN network demonstrated the highest accuracy (84.15%), sensitivity (83.96%), and specificity (84.32%) among the three networks. Similarly, I. Shafi et al. [109] highlight the challenge of developing a model for the early detection of lung cancer based on DL-enabled SVM. The model is trained and validated on 888 annotated CT scans from the LIDC/IDRI database, achieving an accuracy of 94% in detecting pulmonary nodules representing early-stage lung cancer. The suggested method outperforms other known approaches, such as complex DL, straightforward ML, and hybrid methods applied to lung CT scans for nodule diagnosis. The results of the study demonstrate the potential for the proposed model to greatly assist radiologists while reducing the need for repeated CT scans and associated radiation exposure. In a recent study, G. Liu et al. [74] aim to improve the accuracy of classifying benign and malignant lung nodules using a novel multimodal attention-based 3D CNN that combines a total of 204 CT scans' features and clinical information. The suggested DL model had an average diagnostic sensitivity for malignant

nodules of 96.2% and an average accuracy for classifying benign and malignant nodules of 81.6%. This outperformed the traditional ResNet and VGG networks, which achieved lower sensitivities and accuracies. M. A. Moragheb et al. [85] improved a hybrid approach for the efficient generation of nodule masks and reduction of false positives in pulmonary or benign nodules with a diameter of 3 cm or less. The approach involves conducting nodule segmentation preprocessing and using a U-Net CNN model for nodule identification. After the training phase on the LIDC IDRI dataset, the U-Net CNN model achieved a dice coefficient of 0.678 and a sensitivity of 75%, reducing false positives from 11.1 to 2.32 false positives per true positive. J. Uthoff et al. [131] examined the benefit of including perinodular parenchymal features in ML tools for pulmonary nodule assessment using a dataset of 363 cases (74 malignant, 289 benign) with pathology-confirmed diagnosis. Radiomic features extracted from the parenchyma surrounding lung nodules were used to build an ANN classifier, which showed improved performance over exclusively nodular features. The best-performing ML tool has features extracted from surrounding parenchyma tissue quartile bands based on nodule diameter, and it achieved 100% of sensitivity and 96% of specificity.

Other studies investigated Sleep-Disordered Breathing (SDB), and Obstructive Sleep Apnea (OSA) to be more precise. In order to detect the severity of SDB, T. Kim et al. [64] aimed to identify acoustic biomarkers that can predict the severity of SDB by analyzing breathing sounds collected from patients during full-night Polysomnography (PSG). The study extracted audio features and selected a group of features that could potentially serve as acoustic biomarkers. Based on the chosen acoustic biomarkers, ML techniques were utilized to classify the severity of SDB. The study has a four-group classification accuracy of 88.3% and a binary classification accuracy of 92.5%. The results suggest that acoustic biomarkers may be used to predict the severity of SDB based on breathing sounds collected using microphones, which could be used outside of specialized facilities for screening purposes. J. Zhang et al. [151] conducted a model for detecting OSA using a single-channel Electrocardiogram (ECG) signal and CNN. The proposed model uses a combination of three types of filters in the first convolution layer, a LSTM to learn temporal dependencies. The model was trained and tested on the Apnea-ECG dataset, which contains 70 ECG records, and achieved a Cohen's kappa coefficient of 0.92, a sensitivity of 96.1%, a specificity of 96.2%, and an accuracy of 96.1% for detecting OSA events. The results indicate that the proposed model outperformed the baseline method and could be a useful tool for detecting OSA based on a single-lead ECG. In another study, A. Sheta et al. [113] propose an automated system for the detection of OSA based on ECG signals. The proposed framework consists of three steps: noise removal using a Notch filter, feature extraction from the ECG signal, and OSA diagnosis using ML and DL models. The Apnea-ECG Database is used in this study. The experimental results demonstrate that the proposed approach achieves an accuracy of 86.25% in the validation stage. On the other hand, H. Elmouaquet et al. [31] present a new deep RNN framework for detecting sleep apnea events automatically from single respiratory channel inputs.

The framework uses LSTM and Bidirectional Long Short-Term Memory (BiLSTM) models for automatic feature extraction and detection of apneic events. The framework is evaluated over three respiratory signals: Oronasal thermal airflow, Nasal pressure (NPRE), and abdominal respiratory inductance plethysmography using PSG data of 17 patients with obstructive, central, and mixed apnea events, which form a total of 45417 segments. The results demonstrate that the proposed framework is effective in detecting apneic events, achieving high true positive rate 90.3%, true negative rate 83.7% and a ROC-AUC of 92.4% using NPRE signal. In another research, M. Sharma et al. [111] investigated a simple and computationally efficient system to detect OSA in elderly subjects using respiratory and oximetry signals. The proposed model is developed using the Sleep Heart Health Study (SHHS) database, which includes two groups SHHS-1 and SHHS-2, involving 5,793 and 2,651 subjects, respectively, with an average age of  $\geq 60$  years. The model is created by combining GentleBoost and Random under-sampling Boosting algorithms. The developed model achieved an accuracy of 89.39% and 84.64% for the imbalanced and balanced datasets, respectively using a 10-fold cross-validation technique. These results show that the respiratory and SpO2 signals-based model can be used for automated OSA detection and can be better than the state-of-the-art models. Lastly, a very recent work from A. I. Sharaf [110], where he proposed an automated approach to detect and classify apneic events in OSA using single-lead ECG signals. The approach involves applying Wavelet Scattering Transformation (WST) to the ECG signals, extracting features from the WST coefficients, and using Random Forest classifier for classification. The experiment resulted in an accuracy of 91.65% and 90.35% using the 10-fold and hold-out cross-validation methods, respectively, outperforming most existing methodologies.

## 2.11 Conclusion

This chapter provided a comprehensive overview of the theoretical background of ML and DL, covering important topics related to algorithms, techniques, and methodologies that have been used to build powerful ML to automatically handle different real-life issues nowadays, medical diagnostic tools, for instance, with a particular focus on audio data, and the possibility to extract and compute different features such as spectral-based, and energy-related, which can yield valuable patterns and enable the development of robust and efficient models in the upcoming chapters of this thesis, which involve a particular focus on building ML and DL model for a supervised task, which is the diagnosis of COVID-19. Finally, exploring the use of ML and DL in the diagnosis of different respiratory diseases such as Pneumonia, Lung cancer, Sleep Apnea, Chronic obstructive pulmonary disease, Pneumothorax, and Tuberculosis, has shown promising results, and greatly motivate our findings, thus, in the next chapter, a detailed literature review of recent works will be presented, for specifically, COVID-19 diagnosis from, cough sound, or, respiratory sounds to be more general.



## LITERATURE REVIEW

### 3.1 Introduction

As shown in the last chapter, ML and DL have become powerful tools for medical diagnosis, particularly in the context of respiratory diseases. With the emergence of COVID-19 outbreak, the need for a similar tool is raised by the research community, with this respect, many works have been done and evaluated. In particular, different types of audio features have been extracted to find potential relevant patterns from cough and respiratory sounds, which can lead to better diagnosis and classification performance. In this chapter, a critical summary of recent works related to diagnosing COVID-19 from cough sound, and respiratory sounds will be provided, as well as a descriptive and statistical overview of the existing datasets, with a particular focus on the one used in the present work. Moreover, we will categorize the related works according to their ML and DL approaches. Through this review, we aim to gain insights into the state-of-the-art methods for COVID-19 diagnosis and to identify potential research directions through a comparative analysis, that can contribute to the development of an effective and efficient diagnostic system. This chapter is an extended and revised version of the literature review presented in our papers [44, 45, 136, 137]. First, the datasets of cough and respiratory sounds under COVID-19 will be presented, then, a deep review of recent ML and DL-based methods and frameworks that are employed to diagnose COVID-19 from audio, a particular focus will be put forward to the employed data augmentation and class balancing due to the lack of COVID-19 positive samples, after that, comparative analysis and discussion, and the important limitations of the methods, followed by a short introduction about our proposed approaches.

## 3.2 Cough and respiratory sounds under COVID-19 data availability

Since the emergence of COVID-19 outbreak, the demand for quick, accurate, and safe diagnostic techniques has been rapidly growing. Cough and respiratory sounds have become a valuable source for diagnosing COVID-19 patients as they can provide essential information about the patient's condition, such as the presence of wheezing, breathing rate, and respiratory effort. Therefore, several datasets have been compiled for the purpose of diagnosing COVID-19 from cough and respiratory sounds, including **COUGHVID** Dataset, **UC** dataset, and **Coswara** dataset. These datasets provide a valuable resource for researchers and practitioners to develop and evaluate ML and DL algorithms for diagnosing COVID-19 from cough and respiratory sounds. A short descriptive and statistical overview of each one is given below:

### 3.2.1 COUGHVID dataset

COUGHVID [91] is a dataset consisting of audio recordings of cough sounds. The dataset was created by L. Orlandic et al. from the École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, to aid in cough sound analysis and ML models development for COVID-19 screening. Table 3.1 shows the overall statistics of the provided recordings. It has been collected through a web application from April 1<sup>st</sup>, 2020 until December 1<sup>st</sup>, 2020 where users across the world were asked to click on the recording button, then fill out a meta-data questionnaire that contains age, gender, geolocation information, previous existing respiratory conditions, and COVID-19 status. The reported COVID-19 status consists of three possible values or classes *Healthy*, *COVID-19*, and *Symptomatic*. In order to clinically validate the crowdsourced data, it was annotated by four physician experts who provided more than 1000 annotations for various items including the quality and type of cough, as well as audible dyspnea, wheezing, stridor, choking, and nasal congestion. In addition, the reports of physicians include impressions about the patient's infection and the severity of their condition. The audio recordings codec is Opus, with a sample rate of 48kHz. The dataset is publicly available in Zenodo <sup>1</sup> platform and has been used in various studies for COVID-19 diagnosis using data mining and ML methods. COUGHVID has been used to build and test the effectiveness of all ML and DL which are reported in this thesis. Further focus on COUGHVID dataset will be carried out in the next chapters in the data engineering steps.

### 3.2.2 UC dataset

The UC dataset is a crowdsourced data collection framework that involves a web-based app and an Android app for data gathering [19]. Users input their symptoms and record respiratory

---

<sup>1</sup><https://zenodo.org/record/4048312>

	Healthy	Symptomatic	COVID-19	No Status	<b>Total recordings</b>
Number of samples	12479	2590	1155	11326	<b>27550</b>

Table 3.1: COVID-19 status label distribution over the three statuses *Healthy*, *COVID-19*, and *Symptomatic*, which are reported by users

sounds, including coughing and deep breathing. The dataset contains information about age, gender, medical history, COVID-19 testing, and location. In addition, the Android and iOS apps prompt users to input further sounds and symptoms every two days, allowing for the study of the progression of user health based on sounds. By May 2020, the dataset is composed of 378 unique users collected from web and Android apps, with a total sample count of 459. The dataset contains information about users who tested positive for COVID-19 and those who did not. The age distribution in the dataset is skewed towards middle age. The dataset provides 32 cases who have declared COVID-19 positive status. A newer version with more samples is reported in [146].

### 3.2.3 Coswara dataset

The Coswara dataset [112] is a collection of respiratory sounds, including cough (shallow and heavy), breath (deep and shallow), and voice (vowels o/e/a, counting normal, and fast). It was widely used for the development of a diagnosis tool for COVID-19. The dataset was collected via worldwide crowdsourcing using a web app, with participants providing metadata and recording sound samples using their device microphone. As of August 7, 2020, the dataset has a participant count of 941, with each participant providing 9 audio files, one for each of the sound categories. The first released version of Coswara dataset consists of 6507 clean, 1117 noisy, and 845 highly degraded audio files. Table 3.2 shows summarized statistics of the provided recordings.

	Healthy	Not-healthy	<b>Total</b>
Participants	790	104	<b>894</b>
Number of samples	5947	560	<b>6507</b>

Table 3.2: Coswara status label distribution over COVID-19 statuses *Healthy*, and *Not-healthy*, by excluding the noisy and highly degraded audio recordings

Overall, these are the most used datasets for COVID-19 screening based on respiratory sounds, where they have been crowdsourced through web and mobile apps, also, they provided some additional meta-data that can be beneficial for building a demographic or clinical information-based diagnosis system. In terms of the number of Non-healthy of COVID-19 cases, COUGHVID provides more samples compared to the first released version of UC and Coswara. However, we shall note that all of these datasets suffer from class imbalance issue, where the number of COVID-19 positive cases is relatively small compared to healthy and symptomatic cases,

Figure 3.1 shows a histogram of the class distribution of discussed datasets, where it clearly shows a non-balanced distribution. This class imbalance can decrease the performance of ML models and may require special handling during data engineering and model training.

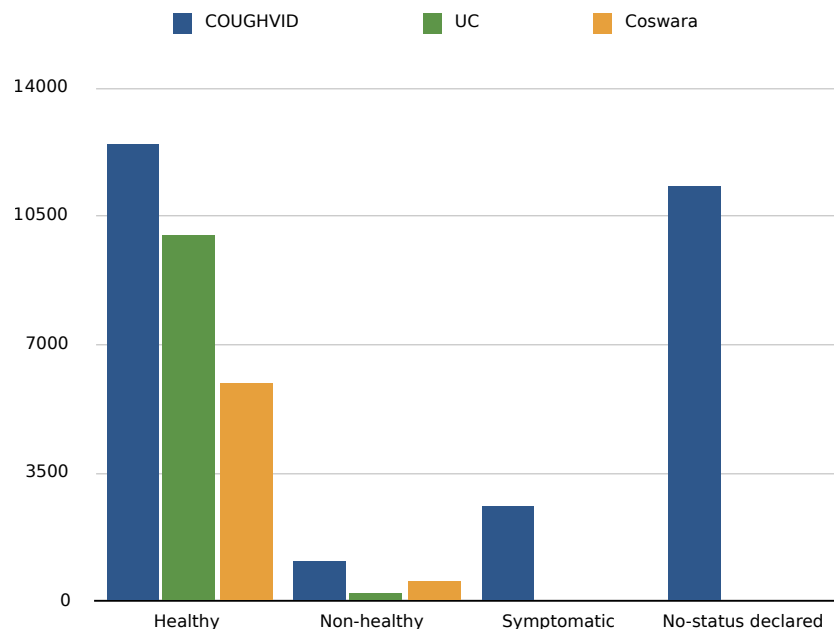


Figure 3.1: Histogram for the class label distribution of the three datasets *COUGHVID*, *UC*, and *Coswara*.

### 3.3 Related works

As part of this literature review, several systematic and technical reviews have been collected and analyzed, these studies have been conducted by researchers from various fields, including computer science, biomedical engineering, and medicine. In these reviews, several study and methodology categorizations have been followed, K. Santosh et al. [103] categorize the works based on several factors, and the most important was shallow and ML-based learning using *handcrafted features*, and DL architecture to extract deep features from Mel-spectrograms. M. Husain et al. [54], investigated a scoping review, in which they examined and categorized different relevant works based on, mostly, *DL architectures* such as CNN and used features. Another review from S. Ghrabli et al. [37], focused only on DL techniques and architectures. In another review study, Md. Mahadi Hasan et al. [48], categorized their literature search by only ML and DL-based techniques. Lastly, without any specific categorization, K. Kumar et al. [70], reviewed several works, however, they mentioned various methodologies in terms of ML and DL. Overall, we found that almost all of these studies focus on comparing the performance of ML-based and DL-based methodologies for COVID-19 diagnosis, whereas others explore the effectiveness of different audio features. However, we have noticed that there is a lack of emphasis on the data augmentation

strategy, which is a crucial step in improving the performance of ML and DL models under class imbalance limitation which was discussed in Section 3.2.

In order to provide a comprehensive and targeted overview of the current state of research in the field of cough-based COVID-19 detection and diagnosis, we have chosen to analyze and summarize a total of 40 research papers after excluding numerous studies based on specific criteria. We have intentionally omitted pre-print articles, as they have not yet undergone the peer-review process and may lack the rigor and validation of published works. Moreover, we have chosen to exclude studies that focus solely on speech or breath sounds to maintain a consistent focus on cough-based analysis. Furthermore, we have also excluded one specific work that relies solely on the signal envelope and its upper and lower boundaries for the classification of cough sounds, and another one that proposes a theoretical framework.

Building ML or DL models for various applications including COVID-19 detection using cough, breathing, and speech recordings, typically involves a pipeline encompassing several techniques and strategies. As such, different review categorizations can be adopted to analyze and assess the works in this discipline. These categorizations can focus on various aspects of the model-building process, such as feature extraction, data augmentation techniques, model selection, or evaluation metrics. Therefore, in this literature review, we will categorize the studies based on ML-based and DL-based methodologies, as well as studies that are based on a comparative analysis between ML and DL, with a particular focus on the used data augmentation techniques, if existed.

### 3.3.1 ML-based Approaches

In one of the pioneering studies in the field, B. Chloë et al. [19] investigate the potential of utilizing respiratory sounds; coughing and breathing, for diagnosing COVID-19. This study compiled and employed UC dataset where a variety of audio features were extracted, such as signal duration, onset, tempo, period, RMS Energy, spectral centroid, Roll-off frequency, zero-crossing, 13 first components of MFCCs,  $\Delta$ -MFCCs,  $\Delta^2$ -MFCCs, making a total of 477 handcrafted feature. In addition, 256 VGGish features were extracted and combined with handcrafted features. Dimensionality reduction was conducted using PCA. The results demonstrated that SVM using RBF as kernel and PCA could effectively distinguish between COVID-19 and non-COVID-19 users, with precision and recall rates of 80% and 72%, respectively, and a ROC-AUC of 82%. The study also analyzed the impact of different sound modalities and dimensionality reduction on performance as well as the impact of the employed data augmentation in results improvement. The findings indicated that combining cough and breathing sounds improved performance, and data augmentation (Amplifying the original signal, Adding white noise, and Changing pitch and speed) significantly boosted recall and ROC-AUC to 90% and 87%, respectively.

In the study conducted by J. Vrindavanam et al. [135], various spectral and pitch handcrafted features were extracted (Dominant Crest factor, Linear Predictor Coefficients (LPC), Dominant Frequency, Power Spectral Density, RMS Energy, Zero-cross rating, Formant Frequency, Spectral

Centroid, Spectral Bandwidth, Spectral Roll-off, Spectral Flatness) from cough audio samples which are collected from UC dataset, Coswara, and Freesound for healthy samples, followed by a feature selection process based on importance using methods such as Information Value (IV), Random Forest Feature importance, RFE, Extra Trees Classifier (ETC), Chi-Square value (CS), and L1 Feature selection (L\_ONE). The researchers evaluated Logistic Regression, SVM, and Random Forest classifiers using GridSearch. The best performance was achieved by SVM, with an accuracy of 83.9% and a sensitivity of 81.2%. This study highlights the potential of using classical ML models, along with handcrafted feature and selection techniques, to diagnose COVID-19 patients based on solely cough sound.

In another study, N. M. Manshouri [81] employed SVM with radial basis function kernel. First, MFCCs, and Power Spectral Density features have been extracted using the waterfall representation of the signal frequency spectrum. The coughs of individuals with COVID-19 were identified with a 95.86% classification accuracy. The detection of COVID-19 coughs was accomplished with a sensitivity of 98.6% and a specificity of 91.7%.

A. Tena et al. [128] conducted a study on the automated identification of COVID-19 from cough sounds by analyzing time-frequency features extracted from various datasets, including the University of Lleida (UL), UC, Pertussis dataset, Coswara dataset, and Virufy dataset. They used YAMNet to identify cough boundaries within the audio files and then transformed the cough samples into time-frequency representations using Wigner distribution (WD) and Choi-Williams exponential function. The study extracted 39 time-frequency features by dividing the spectrum into seven frequency bands, calculating various attributes such as instantaneous spectral energy, instantaneous frequency, instantaneous frequency peak, the mean frequency of the total spectrum, joint, instantaneous Shannon entropies, spectral Shannon entropies, Kurtosis, and moments of the marginal signals of instantaneous power and Spectral density. Two feature selection strategies were employed: RFE and Autoencoder. The study conducted several tasks, in the one related to COVID-19 Positive versus Negative classification, ML models, including Random Forest, SVM, LDA, and Logistic Regression are employed. When employing RFE and Random Forest, the study achieved an accuracy of 89.79%, a sensitivity of 93.81%, and an AUC of 96.04%. In contrast, using the Autoencoder and RF, they reported an accuracy of 83.67%, a sensitivity of 89.58%, and an AUC of 93.56%.

In another study [86], P. Mouawad et al. explore the classification of COVID-19 using cough and sustained vowel sounds. The authors addressed the imbalance in their dataset by employing a novel method of informative undersampling using an information-theoretic measure of information rate, combined with RUS + IR. They extracted symbolic recurrence quantification measures along with Mel Frequency Cepstral Coefficients (MFCC) features from the audio signals, which captured complex dynamics in vocal sounds. For classification, they utilized the XGBoost model and optimized performance using a threshold-moving method. The results demonstrated that the proposed method achieved an F1-score of 91% for cough sounds and 89% for sustained vowel

sounds. This research highlights the potential of using symbolic recurrence quantification measures and MFCC features in effectively discriminating between healthy and COVID-19-affected individuals based on cough and sustained vowel sounds.

I. Södergren et al. [117] proposed a robust ensemble method, by utilizing the ComParE 2016 feature set, which yields a feature vector of size 6373, and ensembling two classical ML models: SVM and Random Forest. The average of the weighted predictions from both models is taken to create an ensemble that demonstrated strong performance on the DiCOVA challenge data. The ensemble model achieved a ROC-AUC score of 85.21%, outperforming the baseline provided by the challenge organizers and placing fourth on the leaderboard. The Random Forest method, in particular, was found to converge toward high prediction performance with consistent prediction correlation, despite randomly initiating feature selection. These results highlight the effectiveness of combining classical ML models, such as SVM and Random Forest, with the ComParE 2016 feature set for the task of detecting COVID-19 from cough sounds.

In order to examine the performance of state-of-the-art ML methods in effectively classifying COVID-19 from cough sounds, N. K. Chowdhury et al. [25] utilized four datasets: UC dataset and Coswara for training, Virufy, and NoCoCoDa for validation. In this study, a set of hand-crafted features have been used: MFCCs, Mel-Scaled Spectrogram, Tonal Centroid, Chromagram, and Spectral Contrast. An ensemble-based multi-criteria decision-making (MCDM) approach was proposed to identify top-performing ML techniques among Extremely Randomized Trees (Extra-Trees), SVM, Random Forest, Adaptive Boosting (AdaBoost), MLP, Extreme Gradient Boosting (XGBoost), Gradient Boosting (GB), Logistic Regression, k-Nearest Neighbors (kNN) and Histogram-based Gradient Boosting (HGBoost). The MCDM method integrates ensemble methods, employs the Technique for Order Preference by Similarity to the Ideal Solution (TOPSIS) for ranking purposes, and then, utilizes entropy to determine evaluation criteria weights. The class imbalance issue was addressed by employing SMOTE during training. Three different training strategies have been explored, including nested Cross-Validation with hyper-parameter optimization. RFE with Cross-Validation is used for feature reduction under various estimators. Experiments evaluations show that using Extra-Trees classifier, the model achieved a ROC-AUC of 95%, precision of 100%, and sensitivity of 97%.

In a cough analysis study by J. J. Valdés et al. [132], different cough types (Dry, Wet, Whooping, and COVID-19) have been analyzed using supervised and unsupervised methods using a small dataset. Supervised classifiers, kNN, XGBoost, and Random Forest, were applied to input dissimilarity spaces computed using Dynamic Time Warping (DTW) measures. The results demonstrated high performance in identifying COVID-19 coughs, with sensitivities of 0.71 for kNN and 0.86 for both XGBoost and Random Forest. The presented results are considered preliminary, due to the dataset size, absence of signal preprocessing techniques, and hyperparameter tuning, which suggest more improvement and refinement.

In another comparative study between different ML approaches, F. Z. Solak [118] investigated

the use of Virufy dataset. Due to the class imbalance of this dataset, the ADASYN oversampling technique was employed for data balancing. Through a non-linear analysis (Entropy measures, Lempel-Ziv Complexity (LZC), and Multifractal Detrended Fluctuation Analysis), the author extracted 29 features. The most 9 effective features were then selected using the ReliefF method. Five algorithms SVM with Radial Basis Function (SVM-RBF), Random Forest, AdaBoost, ANN, and kNN were utilized to classify cough sounds as COVID-19 or Non-COVID-19. The study found that the SVM-RBF classifier, combined with the selected features, achieved a 95.8% classification accuracy, 93.1% sensitivity, 98.6% specificity, 98.6% precision, a Kappa value of 0.92, and a 93.2% ROC-AUC.

A hybrid framework named *CR19*, proposed by E. E. Hemdan et al. [50], leverages the genetic algorithm in combination with ML techniques to enhance its accuracy. The performance of the hybrid framework (GA-ML) is compared to standalone ML approaches; Logistic Regression, LDA, kNN, Classification And Regression Tree (CART), Naive Bayes, and SVM. The results demonstrate that the hybrid GA-ML technique outperforms the standalone ML methods based on different evaluation metrics. The proposed framework achieves an accuracy of 92.19%, 94.32%, 97.87%, 92.19%, 91.48%, and 93.61%, compared to accuracies of 90.78%, 92.90%, 95.74%, 87.94%, 81.56%, and 92.198% for Logistic Regression, LDA, kNN, CART, Naive Bayes, and SVM, respectively.

Similarly, based on Particle Swarm Optimization (PSO), M. A. A. Albadr et al. [4], proposed PSO-Extreme Machine Learning (PSO-ELM), which is a ML algorithm known for its accuracy and speed in classification tasks. The proposed system uses MFCCs as input features, which were extracted from respiratory system voice samples obtained from the Corona Hack Respiratory Sound Dataset (CHRSD). The system encompasses thirteen different scenarios: breath deep, breath shallow, all breath, cough heavy, cough shallow, all cough, count fast, count normal, all count, vowel a, vowel e, vowel o, and all vowels. Experimental results reveal that the PSO-ELM classifier achieves the highest accuracy in detecting COVID-19 across all scenarios, reaching accuracies of 95.83%, 91.67%, 89.13%, 96.43%, 92.86%, 88.89%, 96.15%, 96.43%, 88.46%, 96.15%, 96.15%, 95.83%, and 82.89% for breath deep, breath shallow, all breath, cough heavy, cough shallow, all cough, count fast, count normal, all count, vowel a, vowel e, vowel o, and all vowel scenarios, respectively.

In another study, M. Melek [80] used Virufy and NoCoCoDa datasets and extracted MFCCs features from cough sounds, which are then classified with seven different ML classifiers; SVM, LDA, kNN, and Partial Least-Squares Regression (PLSR). LOOCV strategy is implemented to determine optimal hyperparameter values for MFCCs and classifiers. kNN classifier based on the Euclidean distance achieved an accuracy of 0.9833 after applying Sequential Forward Selection (SFS) to identify the most effective features for each classifier.

Table 3.3 provides a summary of the ML-based approaches discussed in this section, highlighting the dataset names, extracted and used features, learning and classification methodology,



data augmentation and class balancing techniques, and the main reported results.

Table 3.3: Summary of ML-based studies. DA refers to Data Augmentation, CB refers to Class Balancing, dash (-) refers to not reported (used and not mentioned, or not completely used), *Acc* for accuracy, *Rec* for recall, *AUC* for ROC-AUC, *Prec* for precision, *Spec* for specificity

Work	Data	Features	ML model	DA and CB	Results
B. Chloë et al. [19]	UC	477 handcrafted such as MFCCs and RMS Energy. 256 features extracted using VGGish	SVM with RBF Kernel and PCA	<b>DA:</b> applied (Amplifying the original signal, Adding white noise, Changing pitch and speed). <b>CB:</b> -	<i>Rec:</i> 90% <i>Prec:</i> 70% <i>AUC:</i> 87%
J. Vrindavanam et al. [135]	UC Coswara Freesound	Dominant Crest factor, LPC, Dominant Frequency, Power Spectral Density, RMSE, ZCR, Formant Frequency, Spectral Centroid, Bandwidth, Roll-off, and Flatness.	SVM	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 83.9% <i>Rec:</i> 81.2%
N. M. Manshouri [81]	-	19 MFCCs Power Spectral Density	SVM	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 95.86% <i>Rec:</i> 98.6% <i>Spec:</i> 91.7%
A. Tena et al. [128]	UL UC Pertussis Coswara Virufy	39 time-frequency features	Random Forest SVM LDA Logistic Regression	<b>DA:</b> - <b>CB:</b> -	RFE <i>Acc:</i> 89.79% <i>Rec:</i> 93.81% <i>AUC:</i> 96.04%
P. Mouawad et al. [86]	Private	MFCC, Symbolic Recurrence Quantification measures	XGBoost	<b>DA:</b> - <b>CB:</b> applied (Informative undersampling with RUS + IR)	F1-score: 91% (cough)
I. Södergren et al. [117]	DiCOVA	ComParE 2016 feature set (6373 features)	Ensemble (SVM, Random Forest)	<b>DA:</b> - <b>CB:</b> -	<i>AUC:</i> 85.21%

Work	Data	Features	ML model	DA and CB	Results
N. K. Chowdhury et al. [25]	UC Coswara Virufy NoCoCoDa	MFCCs, Mel-Scaled Spectrogram, Tonal Centroid, Chromagram, Spectral Contrast	Ensemble-based MCDM: Extra-Trees, SVM, Random Forest, AdaBoost, MLP, XGBoost, GBoost, Logistic Regression, kNN, HGBost	<b>DA:</b> - <b>CB:</b> applied (SMOTE)	<i>Rec:</i> 97% <i>Prec:</i> 100% <i>AUC:</i> 95%
J. J. Valdés et al. [132]	Private	DTW measures	kNN, XGBoost, Random Forest	<b>DA:</b> - <b>CB:</b> -	<i>Rec:</i> 0.86 (XGBoost and Random Forest)
F. Z. Solak et al. [118]	Virufy	29 non-linear features, ReliefF feature selection	SVM-RBF, Random Forest, AdaBoost, ANN, kNN	<b>DA:</b> - <b>CB:</b> applied (ADASYN oversampling)	<i>Acc</i> 95.8%, <i>Rec:</i> 93.1%, <i>Prec:</i> 98.6%, <i>Spec</i> 98.6%, Kappa: 0.92, <i>AUC:</i> 93.2%
E. E. Hemdan et al. [50]	Coswara	-	Logistic Regression, LDA, kNN, CART, Naive Bayes, SVM, Hybrid GA-ML framework	<b>DA:</b> - <b>CB:</b> -	GA-ML <i>Acc:</i> 92.19%
M. A. A. Albadr et al. [4]	CHRS D	MFCCs	PSO-ELM	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 96.43% (heavy cough)
M. Melek [80]	Virufy NoCoCoDa	MFCCs	SVM, LDA, kNN, PLSR	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 98.33% (kNN Euclidean)

### 3.3.2 DL-based Approaches

K. K. Lella et al. [71], developed a Deep Convolutional Neural Network (DCNN) model for diagnosing COVID-19 using respiratory sounds from the UC dataset. The study employed three multi-feature channels (De-noising Auto Encoder, Gamma-tone Frequency Cepstral Coefficients filter bank, and Improved Mel-frequency Cepstral Coefficients) to extract deep features from the input sounds. By combining breath, cough, and voice sounds, the DCNN model achieved approxi-

mately 95% accuracy on Task II of the dataset. Four steps of data augmentation techniques have been used: time stretching, two phases of pitch shifting, compression of the range dynamically, and adding background noise, to enhance the model’s performance. This work outperformed previous approaches in testing accuracy for the same task for this dataset.

A. Imran et al. [55] use a limited dataset of cough sounds from a small number of patients for each of the four groups (Healthy, COVID-19, Pertussis, and Bronchitis). The study proposes a cough classifier based on CNN which receives Mel-spectrogram as input. The cough classifier is trained on ESC-50 dataset in order to classify a recording as cough/non-cough. Furthermore, three different classifiers have been trained to diagnose COVID-19, Deep Transfer Learning-based Multi Class classifier (DTL-MC), Classical Machine Learning-based Multi Class classifier (CML-MC) which is SVM, and Deep Transfer Learning-based Binary Class classifier (DTL-BC). DTL-MC and DTL-BC use MEL-spectrogram as input, while CML-MC (SVM) model uses MFCCs and features extracted using PCA. Additionally, a mediator-based architecture is designed to minimize the probability of misdiagnosis by complementing the weakness of one classifier with the strength of the other and vice versa. The study reports an accuracy of 95.80% for cough detector, 92.64% for DTL-MC, 88.76% for CML-MC, and 92.85% for DTL-BC on the validation set.

In the aim of participation in the DiCOVA challenge, J. Harvill et al. [47] developed a method for classifying COVID-19 from coughing sounds by leveraging unsupervised learning with Auto regressive Predictive Coding (APC) for pre-training on the COUGHVID dataset and fine-tuning on the DiCOVA challenge dataset [88]. Four LSTM layers were used in the APC process to minimize the MSE and subsequently copied the first two layers into the fine-tuning network, using the output of the second layer as extracted features. The fine-tuning network comprised the output from the second layer of the APC model followed by two Bi-directional LSTM layers. SpecAugment spectral augmentation technique was applied during the fine-tuning phase, experimenting with different probabilities of applying it (0%, 50%, and 100%). Experiments have demonstrated that using SpecAugment 100% of the time during fine-tuning slightly outperformed 50% and significantly improved over not using it at all (0%). The proposed configuration achieved a ROC-AUC of 76.81% on validation data and 85.35% on the blind DiCOVA challenge data, ranking third out of 29 participants. The results highlight the effectiveness of APC pre-training and significantly, spectral augmentation in improving the performance of the proposed model.

H. Xue et al. [147] combined the Coswara and UC datasets and introduced a method for self-supervised representation learning using contrastive pre-training on unlabeled data. First, log-compressed mel-filterbanks have been extracted, then, random masking has been employed, which enabled the Transformer structure (feature encoder) to learn representations. In the downstream phase, the feature encoder was fine-tuned with labeled data. The researchers experimented with various models, including VGGish, Gated Recurrent Unit (GRU), GRU-CP (CP: Contrastive Pre-training enabled), Transformer, Transformer-CP, and an ensembling method that combines these methods. During the downstream phase, they tested two similarity functions

*Cosine* and *Bilinear* along with different masking rates (0%, 25%, 50%, 75%, and 100%). The optimal results were achieved by ensembling two Transformer-CPs with varying masking rates, which led to an accuracy of 84.43%, a sensitivity of 73.24%, and a ROC-AUC of 90.03%.

Another interesting research by S. Rao et al. [99], where they employed log-mel spectrogram from DiCOVA 2021 dataset alongside COVID-19 positive samples from the COUGHVID dataset due to the lack of positive samples. The researchers investigated various DL methods, including fully connected neural networks, pruned neural networks based on the Lottery Ticket Hypothesis, and CNNs, specifically, the fine-tuned VGG13 architecture. A crucial aspect of this work involved the use of data augmentation, which is, as above-mentioned, up-sampled positive samples by 400 recordings from COUGHVID dataset. Moreover, the audio mix-up technique was also investigated to avoid possible overfitting by randomly mixing a pair of inputs  $x_1$  and  $x_2$  and their corresponding outputs  $y_1$  and  $y_2$  to enhance model generalization. Mix-up uses a parameter  $\lambda \in (0, 1)$  to create a convex combination of the pair. The custom VGG13 was trained with two configurations, first with BCE, and the second with Focal loss which is commonly used to address the class imbalance issue. Experiments show that ensembling the two configurations along with data augmentation techniques led to a validation ROC-AUC of 82.23% and a test ROC-AUC of 78.3% with a sensitivity of 80.49%, outperforming the fully connected network method and other recent baselines on the DiCOVA challenge.

S. Kim et al. [63], proposed a novel model that combines ResNet-50 and DNN. After hand-crafting MFCCs,  $\Delta$ -MFCCs,  $\Delta^2$ -MFCCs, spectral contrast, chroma, onset, RMS energy, spectral bandwidth, centroid, flatness, roll-off, and ZCR, Bhattacharyya distance was computed for each feature to identify the most effective ones for COVID-19 detection. The model employed two branches: ResNet-50, which processed Mel-spectrogram images, and a DNN that received the selected feature vector. Cough sound data from three crowdsourced datasets UC, Coswara, and COUGHVID were used for training and testing the model. Audio recordings containing only cough sounds were carefully selected through direct listening and inspection. The outputs of these branches were concatenated and forwarded to a fully connected layer for learning and classification. The model demonstrated high sensitivity and specificity rates of 0.95 and 0.96, respectively, indicating improved performance compared to some other studies.

Similarly, in a recently published study, S. Ulukaya et al. [130], propose a multi-branch CNN architecture called MSCCov19Net. It consists of three branches-architecture that combines neural features extracted from diverse domains: MFCC features, Spectrogram images, and Chroma features (Chromagram). The study used two datasets for training: Coswara and COUGHVID, where used Virufy and NoCoCoDa for testing, with a total of 2960/370/370 cough segments for training/validation/testing, respectively. The study employs signal data augmentation strategy by using pitch shifting and white noise addition to enhance the model's generalization. The proposed system shows an accuracy of 61.5% in Virufy data and 90.4% in NoCoCoDa data.

One such study from Z. Ren [100], where he proposed an attention-based ensemble learning

approach that leverages complementary representations from cough samples. Using DiCOVA challenge dataset, the methodology involves extracting three single-model representations from hand-crafted features (log Mel, MFCCs, and ComParE feature set), deep image-from-audio-based representations (Mel-spectrogram images), and deep audio-based representations (extracted from fine-tuned CNN14\_16k and ResNet38 where log Mel-spectrograms are forwarded as input). The attention mechanism was examined at both the feature and decision levels. The results revealed that feature-level and decision-level attention-based fusions outperformed single-model classifiers and max/average fusion for COVID-19 recognition, with the attention-based fusion achieving the best ROC-AUC of 77.96% on the test set.

With the aim of focusing on the interpretability, A. Pal et al. [94] proposed a novel, interpretable COVID-19 diagnosis framework based on DL, using cough sound features and symptoms metadata. The used dataset consists of 30000 audio segments from 150 patients, representing four cough classes (COVID-19, Asthma, Bronchitis, and Healthy). DL technique employs a three-layer DNN model to generate cough embeddings from handcrafted features (MFCCs, log energy, ZCR, Skewness, Entropy, Formant frequencies F1-F4, Kurtosis, and Fundamental frequency  $F_0$ ), while a transformer-based self-attention network called TabNet generates symptom embeddings. The results demonstrate the model's ability to accurately classify between COVID-19 positive and negative cases using cough features, demographic and symptom data, and both types of data combined. Furthermore, the model effectively classifies among the four cough classes, including Bronchitis, Asthma, COVID-19 Positive, and COVID-19 Negative. With a specificity and accuracy of  $95.04 \pm 0.18\%$  and  $96.83 \pm 0.18\%$  respectively, the proposed framework combines the high accuracy and generality of deep neural networks with TabNet's interpretability.

For the same purpose. H. Li et al. [72] examined the data variance between COVID-19 subjects and healthy individuals. Data augmentation techniques; pitch and time shifting, and noise injection have been employed to increase training data variance of Coswara dataset, then convert audio signals into Mel-spectrograms images. A ResNet34 pre-trained model with ImageNet weights was used for automatic feature extraction. The authors reported that most handcrafted features such as MFCCs and RMSE exhibit minor differences between COVID-19 subjects and healthy individuals, with more pronounced differences in speech samples. The system is evaluated on 172 COVID-19 subjects and 793 healthy subjects, achieving a ROC-AUC of over 70%.

In another transfer learning-based study, M. Aly in [8], proposed a two-phase model for COVID-19 detection using cough and breath sounds. The first phase involves Mel-spectrogram, while the second phase focuses on feature extraction and classification with nine pre-trained models; ResNet18/34/50/100/101, GoogLeNet, SqueezeNet, MobileNetv2, and NASNetMobile. The dataset, derived from the Coswara dataset, comprises data from approximately 1600 individuals (1185 males and 415 females). Data augmentation was employed, though the specific technique was not explicitly detailed. The classification model achieved remarkable accuracy of 99.2% with

the SGD with Momentum (SGDM) optimizer. Among tested models, ResNet18 emerged as the most stable for classifying cough and breath sounds from a limited dataset, demonstrating a sensitivity of 98.3% and a specificity of 97.8%.

Building on the previous study [8], another model has been introduced by M. Loey and S. Mirjalili [76], in which, the sound-to-image transformation was optimized by the scalogram technique, then, the classification step was performed using six pre-trained models; GoogleNet, ResNet18, ResNet50, ResNet101, MobileNetv2, and NASNetMobile. The dataset employed is based on COUGHVID dataset, which consists of 1457 cough sound samples, with 755 from COVID-19 patients and 702 from healthy individuals. The recognition model achieves an accuracy of 94.9% with the SGDM optimizer. Similar to the previous study, ResNet18 proves to be the most stable model for classifying cough sounds from a limited dataset, yielding a sensitivity of 94.44% and a specificity of 95.37%.

Using different CNN-based architectures, M. Esposito et al. [32], utilize data from Coswara and COUGHVID datasets, to employ several CNN architectures; transfer learning, with VGG13, and custom architectures; Convolutional Recurrent Neural Network (CRNN), Gated Convolutional Neural Network (GCNN), Gated Convolutional Recurrent Neural Network (GCRNN), and a fusion method. As input features, Mel-spectrograms were utilized. Initial results show an accuracy of 61.81%, but more recent findings have improved this figure to 80%.

Another study from S. K. Mahanta et al. [77], which employs ConvNets, which is a CNN model, trained on a subset of Coswara dataset (approximately 1.36 hours of cough sound recordings from 75 COVID-19 positive and 965 COVID-19 negative subjects). To handle class imbalance issue, and increase the model's generalizability, some signal-based data augmentation techniques were applied; Time stretching, Pitch shifting, Time shifting, Trimming, and gaining. MFCCs were extracted to be the input of the proposed ConvNets. By achieving a ROC-AUC score of 87.07% on a blind test set, this result surpasses the DiCOVA 2021 Challenge's baseline model by 23%, securing the top position on the leaderboard.

In another study, C. Wall et al. [140] used two datasets, ICBHI and Coswara, to create five customized datasets, labeled D1–D5. D2 is a Coswara dataset consisting of only coughing sounds. After extracting MFCCs features, various DL models have been employed: Attention-Convolutional Recurrent Neural Network (A-CRNN), Attention-BiLSTM (A-BiLSTM), Attention-Bidirectional Gated Recurrent Unit (BiGRU) (A-BiGRU), and CNN. Regarding D2, 0.9825 was the highest reported accuracy, using by A-BiGRU model, with a sensitivity of 1.0, and 0.96 of specificity.

In another similar work, Y. He et al. [49] proposed an attention-based network, named TFA-CLSTMNN, which refers to Time-Frequency Attention-Convolutional LSTM Neural Network. MFCCs features were computed and used as neural network input. Experimental results show an accuracy of 95% on publicly available datasets.

V. Dentamaro et al. [29], proposed AUCO ResNet architecture, which is based on ResNet

architecture and includes three attention mechanisms for merging relevant information from activation maps at various levels of the network, the model receives Mel-spectrogram as input and applies the first attention mechanism; Convolutional Block Attention Module (CBAM), which selectively emphasizes informative features. The second attention mechanism is a Squeeze-and-Excitation (SE) block, which recalibrates channel-wise feature responses. The third attention mechanism is the sinusoidal learnable attention mechanism, which captures fine-grained information within the feature maps. The authors re-implemented many state-of-the-art works for fair comparison [19, 26, 55]. The comparison relied on the tasks reported in [19] (Task I. Positive COVID-19 vs. Healthy without symptoms, Task II. Positive COVID-19 reporting cough vs. Healthy, and Task III. Positive-COVID-19 vs. Healthy with cough and asthma). The experimental result shows a significant improvement for Task II compared to the state-of-the-art works, where it has shown an accuracy of 0.9256, a precision of 0.8881, 0.9256 of sensitivity, and 0.9257 of ROC-AUC.

In another work, L. K. Kumar and P. J. A. Alphonse [67] proposed a light-weight CNN model using Modified-MFCCs and Enhanced Gamma-tone Frequency Cepstral Coefficients (GFCCs) to classify various respiratory diseases, including Asthma, Bronchitis, Pertussis, and COVID-19. The primary benchmark dataset was obtained from UC while data related to Pertussis and Bronchitis diseases were collected from different crowdsourced sound datasets. Different receptive fields and depths have been employed and tested in the proposed model to obtain various contextual information for classification. Experiments suggested that a  $1 \times 12$  receptive field and a depth of 5-Layer for the light-weight CNN were optimal for extracting and identifying features from respiratory sound data. The proposed model has shown approximately 4-10% accuracy improvement compared to conventional MFCCs features. The model achieves around 92% accuracy in predicting COVID-19.

T. Rahman et al. [98], presented a stacking CNN model which receives cough or breath spectrogram, followed by a Logistic regression as meta-learner. The combined dataset includes samples from UC and additional crowdsourced data. For symptomatic and asymptomatic patients, the model achieved an accuracy of 96.5% and 98.85% using cough sound spectrograms, respectively. Additionally, it reached 91.03% and 80.01% for breath sound.

In order to develop a diagnosis system that is compatible and suitable for low-power devices, M. Soltanian and K. Borna [119] designed a lightweight CNN which receives MFCCs features from Virufy dataset as input. Quadratic form kernels are used to generalize linear convolution by considering cross-correlations within the receptive field and calculating a second-order convolution with an expanded weight tensor which led to enhancing performance. However, second-order convolution also led to a higher computational cost, with this respect, Kernel separation has been used by breaking down the kernel into separate components which maintain the desired nonlinearity. Experiment results show that the separable quadratic convolution layer outperforms the ordinal layer, where it achieved 97.5%, 95.2%, 100%, and 100% for accuracy, sensitivity,

specificity, and precision, respectively, compared to the classical convolution layer which achieved 95%, 90% for accuracy and sensitivity, respectively.

The subsequent, Table 3.4, offers a summary of the DL-based methods addressed in this section. It emphasizes as well, the used dataset, extracted and employed features, learning and classification techniques, data augmentation and class balancing strategies, and the reported findings.

Table 3.4: Summary of DL-based studies.

Work	Data	Features	DL model	DA and CB	Results
K. K. Lella et al. [71]	UC	De-noising Auto Encoder, Gamma-tone Frequency Cepstral Coefficients filter bank, Improved MFCCs	DCNN	<b>DA:</b> applied (Time stretching, two phases of pitch shifting, compression of the range dynamically, and adding background noise) <b>CB:</b> -	<i>Acc:</i> 95% on Task-II
A. Imran et al. [55]	Limited dataset (Healthy from ESC-50, COVID-19, Pertussis, and Bronchitis)	Mel-spectrogram, MFCCs, PCA	CNN (Cough Classifier), DTL-MC, CML-MC (SVM), DTL-BC	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 92.64% (DTL-MC), 88.76% (CML-MC), 92.85% (DTL-BC)
J. Harvill et al. [47]	COUGHVID DiCOVA	Output from the second layer of APC model	APC pre-training with 4 LSTM layers, fine-tuning with 2 Bi-directional LSTM layers	<b>DA:</b> applied (SpecAugment with 100% chance) <b>CB:</b> -	<i>AUC:</i> 76.81% (validation), 85.35% (DiCOVA blind data)
H. Xue et al. [147]	Coswara UC	Log-compressed mel-filterbanks	Contrastive pre-training, ensembling Transformer-CP model with Random masking	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 84.43%, <i>Rec:</i> 73.24%, <i>AUC:</i> 90.03%



Work	Data	Features	DL model	DA and CB	Results
S. Rao et al. [99]	DiCOVA, upsampled with COUGHVID	Log-mel spectrogram	VGG13, Fully connected networks, Pruned networks	<b>DA:</b> applied (Upsampling and Audio mix-up) <b>CB:</b> -	<i>AUC:</i> 78.3%, <i>Rec:</i> 80.49%
S. Kim et al. [63]	UC Coswara COUGHVID	MFCCs, $\Delta$ -MFCCs, $\Delta^2$ -MFCCs, spectral contrast, chroma, onset, RMS energy, spectral bandwidth, centroid, flatness, roll-off, ZCR	Two branches, ResNet-50 and DNN, concatenated and forwarded to fully connected layer for learning	<b>DA:</b> - <b>CB:</b> -	<i>Rec:</i> 0.95, <i>Spec:</i> 0.96
S. Ulukaya et al. [130]	Coswara and COUGHVID for train. Virufy and NoCoCoDa for evaluation	MFCCs features, Spectrogram images, Chroma features (Chromagram)	Multi-branch CNN (MSCCov19Net)	<b>DA:</b> Pitch shifting, White noise addition <b>CB:</b> -	<i>Acc:</i> 61.5% on Virufy, and 90.4% on NoCoCoDa
Z. Ren [100]	DiCOVA	Log Mel, MFCCs, ComParE feature set, Mel-spectrogram images, deep audio-based representations	Attention-based ensemble learning (CNN14_16k, ResNet38)	<b>DA:</b> - <b>CB:</b> -	<i>AUC:</i> 77.96%
A. Pal et al. [94]	Private	MFCCs, log energy, ZCR, Skewness, Entropy, Formant frequencies F1-F4, Kurtosis, Fundamental frequency F0	DNN (3-layer) + TabNet (transformer-based self-attention network)	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 96.83%, <i>Spec:</i> 95.04%
H. Li et al. [72]	Coswara	Mel-spectrograms	ResNet34	<b>DA:</b> applied (pitch and time shifting, noise injection) <b>CB:</b> -	<i>AUC:</i> 70%

CHAPTER 3. LITERATURE REVIEW

<b>Work</b>	<b>Data</b>	<b>Features</b>	<b>DL model</b>	<b>DA and CB</b>	<b>Results</b>
M. Aly [8]	Coswara	Mel-spectrogram	Different versions of ResNet (18/34/50/100/101), GoogLeNet, SqueezeNet, MobileNetv2, and NasNetmobile	<b>DA:</b> applied (strategy not specified) <b>CB:</b> -	<i>Acc:</i> 99.2%, <i>Rec:</i> 98.3%, <i>Spec:</i> 97.8%
M. Loey and S. Mirjalili [76]	COUGHVID	Scalogram	GoogleNet, ResNet18, ResNet50, ResNet101, MobileNetv2, and NasNetmobile	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 94.9%, <i>Rec:</i> 94.44%, <i>Spec:</i> 95.37%
M. Esposito et al. [32]	Coswara COUGHVID	Mel-spectrograms	VGG13, CRNN, GCNN, GCRNN, Fusion method	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 80%
S. K. Mahanta et al. [77]	Coswara	MFCCs	CNNs	<b>DA:</b> applied (Time stretching, Pitch shifting, Time shifting, Trimming and Gaining) <b>CB:</b> -	<i>AUC:</i> 87.07% (DiCOVA challenge)
C. Wall et al. [140]	Coswara dataset (D2): coughing sounds only	MFCCs	A-CRNN, A-BiLSTM, A-BiGRU, CNN	<b>DA:</b> - <b>CB:</b> -	A-BiGRU: <i>Acc:</i> 0.98 <i>Rec:</i> 1.0 <i>Spec:</i> 0.96
Y. He et al. [49]	Publicly available datasets	MFCCs	TFA-CLSTMNN (Time-Frequency Attention-Convolutional LSTM Neural Network)	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 95%

Work	Data	Features	DL model	DA and CB	Results
V. Dentamaro et al. [29]	UC	Mel-spectrogram	AUCO ResNet (with CBAM, SE block, and sinusoidal learnable attention mechanisms)	<b>DA:</b> - <b>CB:</b> -	Task II <i>Acc:</i> 0.92 <i>Prec:</i> 0.88 <i>Rec:</i> 0.92 <i>AUC:</i> 0.92
L. K. Kumar and P. J. A. Alphonse [67]	UC Pertussis and Bronchitis datasets	MMFCC, EGFCC	Light-weight CNN (1 × 12 receptive field, 5-Layer depth)	<b>DA:</b> - <b>CB:</b> -	COVID-19 task <i>Acc:</i> 92%
T. Rahman et al. [98]	UC and crowdsourced data	Spectrogram	Stacking CNN with Logistic Regression meta-learner	<b>DA:</b> - <b>CB:</b> -	Cough: <i>Acc:</i> 96.5% (symptomatic) <i>Acc:</i> 98.85% (asymptomatic)
M. Soltanian and K. Borna [119]	Virufy	MFCCs	Lightweight CNN with separable quadratic convolution	<b>DA:</b> - <b>CB:</b> -	<i>Acc:</i> 97.5% <i>Rec:</i> 95.2% <i>Spec:</i> 100% <i>Prec:</i> 100%

### 3.3.3 Works involved comparison: ML and DL Techniques

On behalf of the INTERSPEECH 2021 Computational Paralinguistics Challenge, B. Schuller et al. [105], utilized two subsets of UC dataset where the quality of the data was manually checked, and all audio recordings were resampled and converted to 16 kHz and mono/16 bit for the COVID-19 Cough Sub-challenge to establish challenging baselines using various audio feature extraction techniques and toolkits. Feature extraction step included ComParE functional features, BoAW features, deep unsupervised representation learning with the AUDEEP toolkit, and deep feature extraction from pre-trained CNNs using the DEEP SPECTRUM toolkit. For feature learning and classification, they employed CNN, LSTM, and SVM models. By implementing a majority voting system among the best-performing models, this study achieved an Unweighted Average Recall (UAR) score of 73.9% for DEEPSPECTRUM+SVM, 70.6% for ComParE functionals+SVM, and 84.6% for CNN+LSTM RNN.

In another comparative study, E. A. Mohammed et al. [84] explored various approaches, including ensembling a CNN model trained from scratch, VGG16, and Tuned-VGG16 to classify

cough sounds as COVID-19 positive or negative cases. The dataset used in the study comprised 1020 audio recordings. Seven features have been extracted: Mel-spectrum, Chroma, Tonal spectrogram, Power spectrum, MFCCs, and raw data segment. Class imbalance by segmenting each cough recording into non-overlapping coughs to prevent losing cough features when splitting each audio file. The study compared shallow ML models such as Naïve Bayes, Logistic Regression, kNN, Random Forest, SGD, XGBoost, and SVM, to DL, CNN from scratch, the original VGG16 model, and VGG16 with data augmentation. The results indicated that DL models outperformed their shallow counterparts, with CNN models trained from scratch showing the highest performance. The researchers also emphasized the importance of data augmentation in enhancing model performance and suggested the use of ensemble models for more robust classifiers with resistance to overfitting. They achieved 77%, 80%, and 71% for ROC-AUC, precision, and recall, respectively.

Similarly, M. Pahar et al. [92] conducted a comparative study of various ML and DL techniques, including ResNet50 (transfer learning), CNN, LSTM, SVM, Logistic regression, and MLP, to classify COVID-19 positive and negative cases using the Coswara and Sarcos datasets. Input features such as MFCCs, MFCCs with appended velocity, MFCCs with appended acceleration, log energies, ZCR, and Kurtosis are employed in this study. To address data imbalance and augment the data, SMOTE has been used. Transfer learning with ResNet50 achieved a 98% of ROC-AUC and 95% of sensitivity. The study also applied SFS for optimal feature selection, with LSTM demonstrating the best performance when trained on Coswara and tested on Sarcos, achieving a 93.8% of ROC-AUC and 91% of sensitivity.

H. Coppock et al. [26] developed the COVID-19 Identification ResNet (CIdeR), a model based on the ResNet architecture that takes the concatenation of cough and breathing data as input, from Coswara dataset. Spectrograms have been extracted from WAV audio files and log spectrograms for both cough and breathing signals. To address the class imbalance issue, they employed a weighted binary-cross-entropy loss function. For a fair comparison, they used a baseline method involving SVM with ComParE features from OpenSMILE and PCA for selecting the top 100 components with the highest explained variance. The authors carried out four tasks, with the COVID-19 positive vs. COVID-19 negative task yielding the best results for CIdeR in comparison to the baselines. The CIdeR model achieved a ROC-AUC of  $0.846 \pm 0.040$ .

In a recent study, K. A. Nasab et al. [9] compared various supervised ML and DL methods; SVM, Random Forest, Fully Connected neural network, CNN, and LSTM. Cough data have been collected from 40000 individuals, including both healthy and positive cases, using an online web app *sorfeh.com*. To enhance the dataset, two data augmentation strategies have been employed, time-stretching (with an interval of 0.8 to 1.1 times) and pitch-shifting (within a range of -2.5 to 2.5 semi-tones). MFCCs, Chroma, ZCR, RMSE, Spectral Centroid, Bandwidth, Roll-off, Tonnetz, and Mel-spectrogram have been extracted to be used as input to the learning methods, forming a total of 914 features. The models, generally, achieved accuracy, sensitivity, and specificity

rates higher than 70%, with data augmentation proving to be beneficial in improving model performance. LSTM model stood out, reaching an accuracy of 95% and a sensitivity of 95.5%.

In a study conducted by R. Islam [57], two well-known ML and DL techniques were compared for the classification of cough sounds of the Virufy dataset. Chromagram features were extracted from the audio samples and forwarded as input for the classifier algorithms. The authors modeled two ANN based classifiers, specifically CNN and DNN. Experiments demonstrated that the CNN classifier achieved an accuracy of 92.9%, while the DNN reached 91.7% accuracy. Furthermore, the study provided a performance comparison between the proposed system and two popular ML algorithms, SVM and kNN. The comparison shows the performance of DL model CNN against ML models.

By extending their previous work [92], M. Pahar et al. [93] employed seven standard classifiers to achieve a promising baseline performance. Five datasets with a total of 10.29 hours and without COVID-19 labels for pre-training have been included; TASK, Brooklyn, and Wallacedene datasets (Tuberculosis) for cough, Google Audio Set and Freesound datasets for cough, sneeze, speech, and non-vocal sounds, and LibriSpeech for speech data. For classification, three coughing datasets with COVID-19 labels have been used; Coswara, ComParE, and Sarcos. MFCCs,  $\Delta$ -MFCCs,  $\Delta^2$ -MFCCs, linearly-spaced log filterbank energies, and ZCR features have been extracted. Then, CNN, LSTM, have been pre-trained while ResNet50 was fine-tuned to enhance performance. Bottleneck features were extracted from these pre-trained models and used as input for Logistic regression, SVM, MLP, and kNN classifiers. The highest ROC-AUC of 0.98 was achieved with ResNet50 architecture on coughs from the Coswara dataset, while SVM achieved a ROC-AUC of 0.94 and 0.92 on bottleneck features from breaths in Coswara dataset and speech recordings in the ComParE dataset, respectively. Among all vocal audio, coughs carried the strongest COVID-19 signature, followed by breath and speech.

In another comparison between the performance of a ML technique and a DL technique in constructing a low-cost COVID-19 screening tool by D. S. Vijayakumar and M. Sneha [134]. The dataset used in the research is relatively small, comprising 8 COVID-19 cough samples, 28 pneumonia samples, 15 pertussis samples, and 30 normal cough sounds. Feature extraction was carried out using the Mel scale and log-based spectrogram. Two classification models have been built: one utilizing SVM, and the other employing a DL-based LSTM network with an attention mechanism. The results indicate that the SVM classifier achieved an overall accuracy of 94%, while the attention-LSTM classifier reached 88%.

Table 3.5, presents a summary of the comparative studies between ML and DL techniques addressed in this section. It focuses on the utilized datasets, extracted and employed features, compared methodologies, data augmentation and class balancing strategies, and the reported performance metrics.

Table 3.5: Summary of Comparative Studies between ML and DL Techniques

Work	Data	Features	ML & DL models	DA and CB	Results
B. Schuller et al. [105]	UC	ComParE, BoAW, AUDEEP, DEEP SPECTRUM	CNN, LSTM, SVM	<b>DA:</b> - <b>CB:</b> -	<i>UAR:</i> 73.9% (DEEPSPECTRUM+SVM), 70.6% (ComParE+SVM), 84.6% (CNN+LSTM RNN)
E. A. Mohammed et al. [84]	Coswara and other crowdsourced data	Mel-spectrum, Chroma, Tonal spectrogram, Power spectrum, MFCCs, raw data segment	CNN, VGG16, Tuned-VGG16, Naïve Bayes, Logistic Regression, kNN, Random Forest, SGD, XGBoost, SVM	<b>DA and CB:</b> applied (Recording segmentation into non-overlapping coughs)	<i>AUC:</i> 77% <i>Prec:</i> 80% <i>Rec:</i> 71% (Ensembling CNN, VGG16, and Fine-tuned VGG16)
M. Pahar et al. [92]	Coswara Sarcos	MFCCs, $\Delta$ -MFCCs, $\Delta^2$ -MFCCs, log energies, ZCR, Kurtosis	ResNet50, CNN, LSTM, SVM, Logistic Regression, MLP	<b>DA and CB:</b> applied (SMOTE)	ResNet50: <i>Rec:</i> 95% <i>AUC:</i> 98% LSTM (Train on Coswara and test on Sarcos): <i>Rec:</i> 91% <i>AUC:</i> 93.8%
H. Coppock et al. [26]	Coswara	Log spectrograms	CIdeR (ResNet-based), SVM with ComParE features and PCA	<b>DA:</b> - <b>CB:</b> Weighted binary-cross-entropy loss function	CIdeR <i>AUC:</i> 84.6%
K. A. Nasab et al. [9]	Crowdsourced from online web app	MFCCs, Chroma, ZCR, RMSE, Spectral Centroid, Bandwidth, Roll-off, Tonetz, Mel-spectrogram	SVM, Random Forest, Fully Connected Neural Network, CNN, LSTM	<b>DA:</b> Time-stretching, Pitch-shifting <b>CB:</b> -	LSTM: <i>Acc:</i> 95% <i>Rec:</i> 95.5%
R. Islam [57]	Virufy	Chromagram	CNN, DNN, SVM, kNN	<b>DA:</b> - <b>CB:</b> -	CNN: <i>Acc:</i> 92.9%

Work	Data	Features	ML & DL models	DA and CB	Results
M. Pahar et al. [93]	Multiple datasets including Coswara, Brooklyn, and Freesound	MFCCs, $\Delta$ -MFCCs, $\Delta^2$ -MFCCs, Log filterbank energies, ZCR, Bottleneck features from ResNet50	CNN, LSTM, ResNet50, Logistic Regression, SVM, MLP, kNN	<b>DA:</b> - <b>CB:</b> -	ResNet50 and Coswara <i>AUC</i> : 0.98 SVM and Coswara breaths <i>AUC</i> : 0.94 SVM and ComParE speech <i>AUC</i> : 0.92
D. S. Vijayakumar and M. Sneha [134]	8 COVID-19, 28 pneumonia, 15 pertussis, 30 normal coughs	Mel scale, Log-based spectrogram	SVM, LSTM with attention mechanism	<b>DA:</b> - <b>CB:</b> -	SVM <i>Acc</i> : 94% Attention-LSTM <i>Acc</i> : 88%

### 3.4 Comparative analysis

In this section, we will delve into the differences between various methodologies used in the related works, focusing on the performance and characteristics of ML and DL models, the impact of data augmentation strategies, feature extraction and selection methods, and performance metrics.

#### 3.4.1 ML vs. DL

In general, DL models, such as CNN and LSTM, tend to outperform ML models in COVID-19 diagnosis from cough tasks. Table 3.5 shows that various DL techniques and approaches reported in [9, 26, 57, 84, 92, 93, 105]; CNN, LSTM, pre-trained models such as VGG16 and ResNet50, demonstrate higher performance against different ML algorithms; Naïve Bayes, Logistic Regression, kNN, Random Forest, SGD, XGBoost, MLP, and SVM. The studies [9, 47, 92, 105] demonstrated the effectiveness of LSTM in capturing the time-dependent relationships between input features compared to other ML methods. CNNs (including pre-trained models) also have shown promising performance in most of the previous studies [26, 57, 57, 63, 84, 92, 99, 105, 130], where informative and most relevant patterns were efficiently extracted. Attention mechanism (including Transformers) have also shown high performance in [72, 94, 100, 147]. However, there are cases where ML models can achieve competitive results against DL models such as [134], where SVM reported 94% of accuracy against Attention-LSTM which reported only 88%. State-of-the-art review shows some very competitive results using ML algorithms

against DL methods. To present a fair comparison, the same testing data should be considered in both comparison parties, for instance, DiCOVA challenge, ensembling Random Forest and SVM in [117] achieved a ROC-AUC of 85.21%, while Attention-based ensemble of CNN and ResNet38 [100] achieved only 77.96%. Studies like [50, 81, 93, 117] show that SVM method can reach more than 85% of accuracy, sensitivity, or ROC-AUC, also some other studies like [25, 86, 117, 128, 132], demonstrate the effectiveness and the ability of Random Forest, or Tree-based methods, to be more general. ML algorithms can be more effective than DL in some cases, especially when combined with powerful and appropriate feature extraction, feature selection, and also data augmentation steps. The present analysis confirms that even with the development of revolutionary DL techniques, ML algorithms are still able to provide promising results with less computational cost, and faster training, and inference times, compared to DL. However, it is crucial to indicate the limitation of these techniques such as the need for manual feature extraction which can be time-consuming, for instance, computing the large set of energy-related, voicing-related, and spectral-related features, 477 handcrafted features in [19], 6373 features of ComParE in [117]. It may not also capture all relevant information which inevitably led to a decrease in the model's performance [29]. On the other hand, DL techniques may require more computational power and longer training time, and they can also be more complex to explain or interpret, however, these methods work by first, performing feature extraction, and then the classification step, which means there is no need to manually extract audio features [29] where most of the studies used only Mel-spectrogram [55, 72, 99, 100] as model input.

### **3.4.2 Feature extraction and selection methods**

Feature extraction plays a crucial role in building effective ML and DL models for COVID-19 diagnosis from cough sounds. Extracting informative and discriminative features allows the models to capture the unique characteristics of cough sounds associated with COVID-19 and differentiate them from other cough types or healthy coughs. In the studies presented in Tables 3.3, 3.4, and 3.5, various feature extraction techniques have been employed to improve the performance of the models, these include MFCCs [9, 84, 92, 93], chroma features [9, 57, 84], spectral features (Centroid, Bandwidth, Roll-off) [25, 135], and ZCR [9, 63, 92, 93, 135]. These features capture essential information about the cough sounds' frequency, energy distribution, and temporal characteristics, contributing to the classification task's success. Handcrafted feature extraction methods, such as those mentioned above, often require domain knowledge and can be time-consuming and resource-intensive, as mentioned in Section 3.4.1. On the other hand, automatic feature extraction methods using DL architectures like CNN and LSTM, can learn to extract relevant features directly from the time-frequency representation of audio data, without any need for manual intervention. In the studies [8, 29, 55, 76], DL models have shown promising results in automatic feature extraction from Mel-spectrogram, which allows them to build an efficient diagnosis system. However, handcrafted features can still achieve competitive results



when combined with appropriate feature selection and powerful classifiers, as demonstrated in [81, 93, 117]. Furthermore, another type of feature has been widely explored by some studies, which is bottleneck-based representation, which in general, consists of a latent space of an encoder (a compact of the most discriminating features), it has been used in [19] as VGGish, De-noising Auto Encoder in [71], Output from the second layer of APC model in [71], AUDEEP<sup>2</sup> and DEEPSPECTRUM<sup>3</sup> in [105], and Bottleneck features from fine-tuned ResNet50 in [93]. While this type of feature extraction has demonstrated its effectiveness, it can be also considered as feature selection, which is an essential step to improve the efficiency of ML and DL models by reducing the dimensionality of the feature space. Other feature selection techniques have been also investigated and demonstrated their impact on the enhancement of classification models; SFS has been used in [80] and helped to achieve a classification accuracy of 98.33% using kNN algorithm, also in [92] to select the most relevant features to avoid redundancy and possible overfitting. RFE was compared with Autoencoder and has shown a significant impact in boosting Random Forest performance [128] from 83.67% using Autoencoder + Random Forest to 89.79% using RFE + Random Forest, it has been also used in [135] to identify the best 15 features. PCA has been employed in various studies [19, 26, 55] to select the most principal components, as well as LDA [50, 80, 81, 128], however, the latter has been used as a supervised learning method, and not explicitly for feature selection. Feature extraction and selection in COVID-19 cough sound diagnosis face challenges like noise, recording variations, and individual differences. Addressing the quality and relevance of features is essential for accurate and efficient diagnostic models.

### 3.4.3 Data augmentation strategies

As discussed in Section 3.2 and shown in Figure 3.1, all available coughing under COVID-19 datasets suffer from the lack of data, especially positive cases, this was a challenging task due to several factors, the need for social distancing and strict health protocols make it difficult to gather respiratory sounds in environments under control, also the urgency of the pandemic that required quick data collection, which may lead to decrease the quality of collected data, thus, various techniques have been employed to tackle this limitation and make use of the available data efficiently. Data augmentation and class balancing methods have been employed in 14 studies of the 41 we analyzed, with one work that investigated without specifying the strategy. Amplifying original signal, white noise, and changing pitch and speed in [19] reported an improvement of almost 10% in all metrics. The novel Informative undersampling with Information Value in [86] for balancing training data reported a significant difference of 38% in the ROC-AUC metric, as well as other evaluation metrics. SMOTE has been used in [25, 92], to create synthetic samples from minority class, it has shown a performance boosting for various classifiers in *Strategy 1* vs. *Strategy 2* [25], it has been noticed that SMOTE had an impact on almost all classifiers' sensitivity,

<sup>2</sup><https://github.com/auDeep/auDeep>

<sup>3</sup><https://github.com/DeepSpectrum/DeepSpectrum>

which has been improved from 62% to 77% for Random Forest, in [92], several variants of SMOTE has been implemented, however, the best results are still obtained using the standard version. Adaptive Synthetic (ADASYN) oversampling, which is an improved version of SMOTE and works by employing a density distribution as a criterion to determine the number of synthetic samples to be generated for each minority sample, has been used to balance Virufy dataset in [118]. Time stretching, followed by two phases of pitch shifting, compression of the range dynamically, and adding background noise in the form of several types of audio scenes, have been applied sequentially to generate new audio samples [71]. SpecAugment has been used in one work [47], and shown a promising boost of the ROC-AUC from 70.81% to 76.81% in DiCOVA blind data. In [99], DiCOVA challenging has been used for training, and due to the limited number of COVID-19 samples (50), it has been upsampled with 400 samples from COUGHVID data, additionally, Audio mix-up technique has been also used to generate more training samples by randomly mix a pair of samples with a specific parameter  $\lambda$  which ensure the convex combination between the pair. It was mentioned that the employed data augmentation strategy has highly contributed to the model's performance. Pitch shifting and background white noise have been applied in [72, 130] with an additional time shifting in [72]. In [77], five augmentation techniques have been applied, time stretching, pitch shifting, time shifting, trimming, and gaining, where the CNN model showed a high boosting in terms of ROC-AUC, from 72.33% to 87.07%. Another method has been introduced to enrich the training set in [84], which is the segmentation of cough recordings into non-overlapping coughs, this ensured an additional number of samples. Weighted BCE loss function has been used in [26] to handle class imbalance problem. Lastly, time stretching and pitch shifting have been implemented in [9] and show a high-performance boosting of all evaluation metrics, where LSTM model achieved 95.1% instead of 88.83%. Table 3.6 provides a comparison of results with and without data augmentation or class balancing for the works that have reported both findings. This comparison will help to understand the effectiveness of these techniques in enhancing the performance of models for COVID-19 diagnosis from cough sounds.

### **3.5 Identified limitations**

In this literature review, several limitations have been identified that impact the development and performance of COVID-19 diagnosis systems from cough sounds. One significant challenge is the collection of high-quality cough data, which is crucial for creating reliable and accurate diagnostic models. Additionally, a majority of the studies (26 out of 40) lack sufficient COVID-19 samples and do not employ data augmentation and class balancing strategies, which can impact the generalizability of the findings. Furthermore, it is worth noting that a common limitation observed in many studies is the utilization of small datasets, which may limit the robustness and generalizability of the developed models [55, 77, 132, 134]. Another concern is the inadequate consideration of the sensitivity metric, particularly in binary classification-based works (positive

Table 3.6: Comparative results with and without data augmentation or class balancing

Work	Not augmented and not balanced				Augmented or balanced			
	Acc	Rec	Prec	AUC	Acc	Rec	Prec	AUC
[19]	-	0.82	0.80	0.72	-	<b>0.90</b>	0.70	<b>0.87</b>
[86]	0.98	-	-	0.46	0.97	-	-	<b>0.84</b>
[25]	0.85	0.62	0.90	0.81	0.84	<b>0.77</b>	0.75	<b>0.84</b>
[118]	-	-	-	-	0.958	0.931	0.986	0.932
[71]	-	-	-	-	0.95	-	-	-
[47]	-	-	-	0.7081	-	-	-	<b>0.7681</b>
[99]	-	-	-	-	-	-	-	0.8223
[130]	-	-	-	-	0.904	-	-	-
[72]	-	-	-	-	-	-	-	0.70
[77]	-	-	-	0.7233	-	-	-	<b>0.8707</b>
[84]	-	-	-	-	-	0.71	0.80	0.77
[92]	-	-	-	-	-	0.91	-	0.938
[26]	-	-	-	-	-	-	-	0.846
[9]	0.8883	0.9065	0.8752	-	<b>0.951</b>	<b>0.955</b>	<b>0.9453</b>	-

and negative COVID-19 cases). Sensitivity is an important metric in medical diagnosis, especially for a condition like COVID-19, yet it has often been overlooked in the existing literature, with only 81.2% in [135], 73.24% in [147], 80.49% in [99] and 71% in [84]. Moreover, the use of pre-trained models with only small fine-tuned parts of the network may not be sufficient to adapt to the specific task of COVID-19 diagnosis from cough sounds [8, 32, 72, 100]. On the other hand, training and inferencing deep models from scratch can lead to high computational costs, which may be a limiting factor for some research applications [47, 71, 147], though this limitation has been addressed in [67] by suggesting a light-weight CNN, it should have more focus and attention. Addressing these limitations in future research will be crucial for the development of more fast, reliable, and efficient COVID-19 diagnostic systems based on cough sounds.

### 3.6 Statistical summary of the literature review

In this section, we present a visual summary of the related literature through a set of histograms. The first histogram depicts the growth in the number of the collected papers on COVID-19 diagnosis from sound data, starting from 2020. To provide a more detailed overview of the field, two additional subfigures are included. The first subfigure illustrates the distribution of studies based on their methodology, categorizing them into ML-based, DL-based, and comparative studies between ML and DL approaches. The second subfigure focuses on the data augmentation and class balancing techniques employed in these studies, showcasing the number of papers that

utilized Synthetic-based methods, Signal-based augmentation, Spectral-based augmentation, and other strategies. These visualizations offer a comprehensive snapshot of the current state and trends in the research on COVID-19 diagnosis from sound data. Figure 3.2 shows the distribution of published papers over the years, while Figure 3.3 presents the distribution over category, and data augmentation or class balancing strategy.

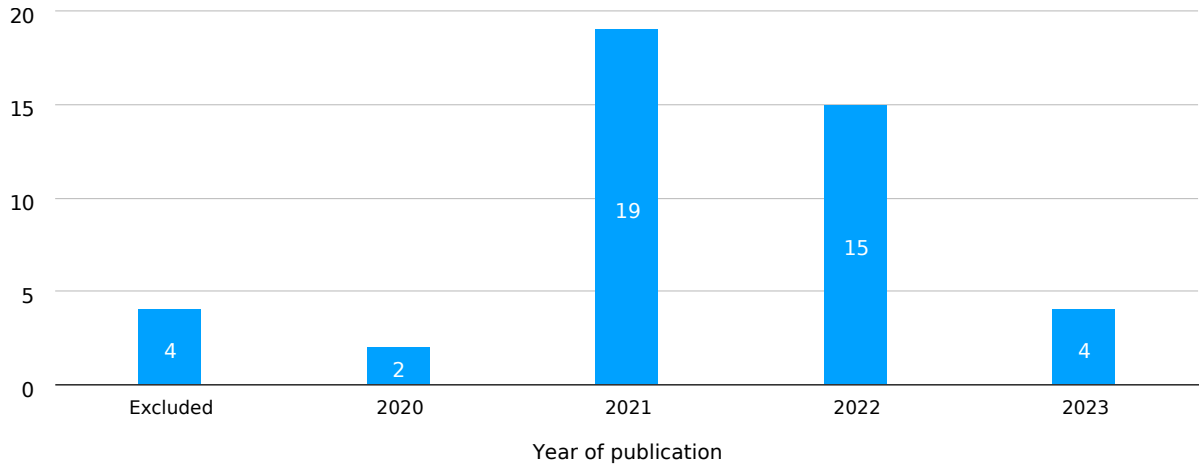


Figure 3.2: Number of published papers, related to COVID-19 diagnosis from respiratory sound and more specifically, cough, per year starting from 2020 until 2023, including the number of excluded which were excluded from the review.

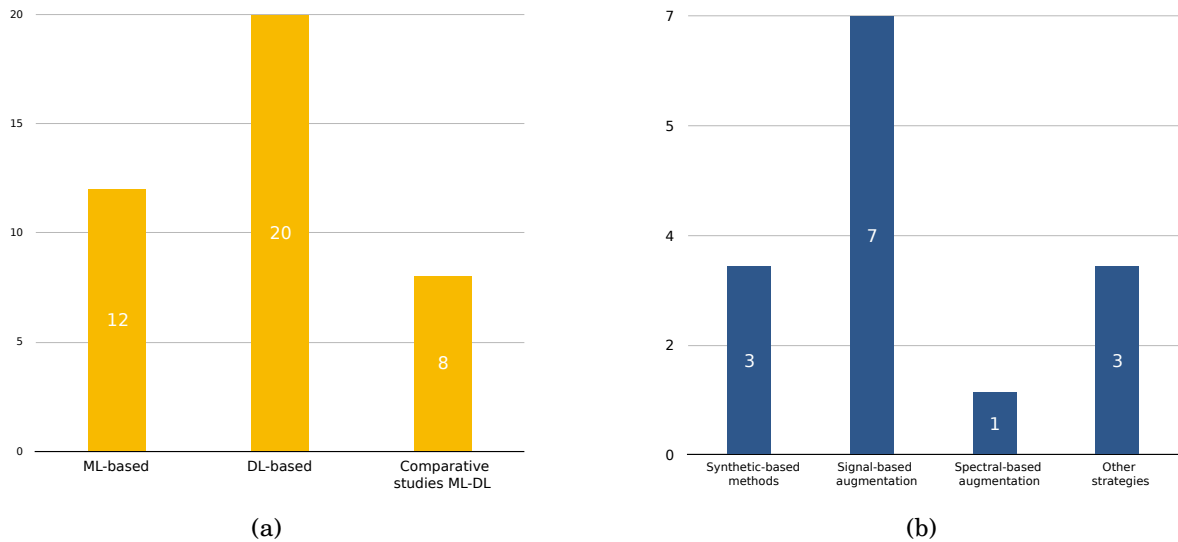


Figure 3.3: (a) Number of published papers per category, ML-based, DL-based, and comparative studies between ML and DL; (b) Number of papers per data augmentation and class balancing type, including synthetic-based methods, signal-based augmentation, spectral-based augmentation, and others.

### 3.7 Conclusion and proposed approaches

In this comparative study, we have analyzed the performance and characteristics of various methodologies used in COVID-19 diagnosis from cough sound. We have focused on ML and DL models, data augmentation strategies, feature extraction and selection methods, and performance metrics. We observed that DL models, such as CNN and LSTM, generally outperform ML models in COVID-19 diagnosis tasks. However, ML algorithms can still achieve competitive results when combined with powerful feature extraction, feature selection, and data augmentation steps. The effectiveness of these techniques can be limited by factors such as manual feature extraction and the need for domain knowledge. Regarding feature extraction and selection methods, we found that extracting informative and discriminative features is crucial for building effective ML and DL models. While handcrafted feature extraction methods can be time-consuming and resource-intensive, automatic feature extraction methods using DL architectures can learn to extract relevant features directly from the time-frequency representation of audio data. Data augmentation and class balancing strategies are essential for addressing the limited availability of cough data, especially the COVID-19-positive ones. Several techniques have been employed in the studies we analyzed, including SpecAugment, time stretching, white noise, and changing pitch and speed, for data augmentation, while using SMOTE and ADASYN and employing novel approaches like Informative undersampling with Information Value for class balancing.

In the upcoming chapters, we will introduce our novel approaches to COVID-19 diagnosis from cough sounds, leveraging both DL and ML techniques. Chapter 4 will discuss our DL approach, which involves a Hybrid CNN-LSTM architecture with Attention mechanism, combined with SpecAugment data augmentation technique to improve performance. In Chapter 5, we will present three distinct ML approaches: 1) Use of Random Forest and Low-Level Descriptors, 2) Random Forest and Stacked Autoencoders, and 3) Enhanced dimensionality reduction with LLE and Random Forest. These methodologies aim to provide an accurate and efficient diagnosis while addressing several of the aforementioned limitations. Firstly, we tackle the issue of the lack of positive COVID-19 samples by investigating the use of different data augmentation and class balancing techniques. This approach aims to create more variability in the dataset and enhance the generalization of our DL and ML models. Additionally, we utilize a larger dataset compared to the majority of existing works, which contributes to the robustness and reliability of our findings. Moreover, we propose fast and efficient ML-based diagnosis systems that do not rely solely on complex DL methods, ensuring easy and fast deployment. Finally, we specifically focus on improving the sensitivity metric, recognizing its significance in medical diagnosis, particularly for COVID-19. By addressing these limitations, our research offers valuable contributions to the development of more effective and reliable COVID-19 diagnostic systems based on cough sounds.

## **Part II**

**Our approaches for diagnosing COVID-19  
from cough sound**

## HYBRID CNN-LSTM WITH ATTENTION-MECHANISM ARCHITECTURE AND SPECAUGMENT FOR COVID-19 DIAGNOSIS FROM COUGH SOUND

### 4.1 Introduction

The state-of-the-art presented in the literature review has provided a strong motivation to investigate novel and more effective approaches for diagnosing COVID-19 from cough sounds. In this chapter, we present an innovative approach based on a hybrid architecture that combines the strengths of CNNs and LSTMs with an attention mechanism. This methodology is inspired and motivated by several works discussed in the literature review, including the use of CNNs [55, 67, 77], LSTMs [9, 47, 49, 105], various attention mechanisms [94, 100, 140, 147], hybridization of different methods [4, 29, 47, 140], and spectral data augmentation [47]. Our approach aims to propose a novel framework that uses signal-based and spectral-based data augmentation to address the lack of positive samples in COUGHVID dataset compared to negative ones and improve the accuracy and reliability of COVID-19 diagnosis using cough sounds using a hybrid DL method. This chapter is a revised and extended version of the methodology presented in our published paper [45]. The subsequent sections cover various aspects of the methodology, including data preprocessing, data augmentation techniques, and the whole pipeline, from data cleaning to attention-based hybrid CNN-LSTM architecture. We also discuss the experimental design, results, and analysis, as well as the potential limitations and future directions of our approach. By leveraging the complementary strengths of CNNs and LSTMs, along with the attention mechanism, our method seeks to provide a robust and efficient solution for diagnosing COVID-19 from cough sounds, potentially contributing to early detection and more effective management of the pandemic.

## 4.2 Materials and methods

Figure 4.1 illustrates an overview of the proposed framework which led to building a model for diagnosing COVID-19 using cough sound data. The framework consists of several interconnected components. Initially, COUGHVID audio dataset recordings undergo a pre-processing phase to ensure the data is clean and prepared for subsequent steps. Following this, two levels of data augmentation are applied to both the audio signal and spectral data, which serve to expand the training set and address the issue of class imbalance. The resulting Mel spectrogram features are then input into a novel attention-based hybrid CNN-LSTM model, which generates binary classification outputs. The model’s performance is assessed using 10-fold cross-validation, during which, Accuracy, Precision, Recall, F1-score, and ROC-AUC evaluation metrics are calculated at each epoch to evaluate the classification performance.

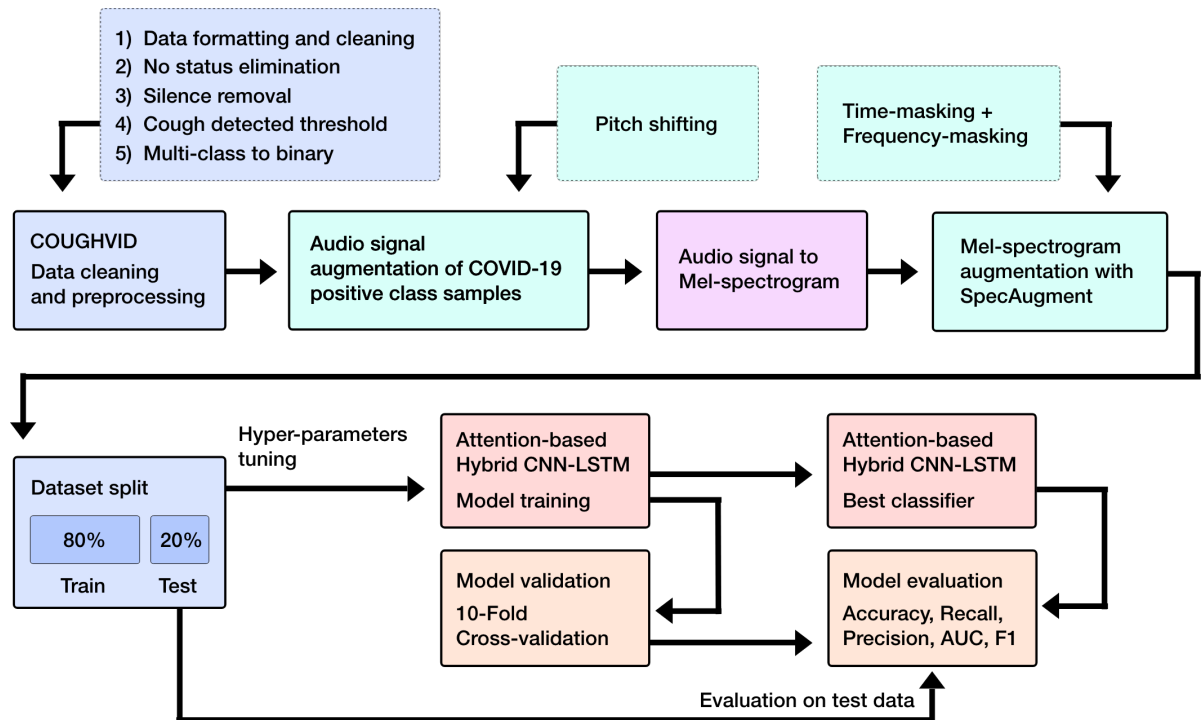


Figure 4.1: Overview of the proposed framework.

The steps mentioned in Figure 4.1 will be described in detail starting from the next section.

### 4.2.1 Dataset description

Briefly, as mentioned in Section 3.2, we used COUGHVID crowdsourcing dataset [91] from Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. This largest publicly available dataset consists of 27550 audio recordings including 1155 positive cases, collected through a web application from April 1st, 2020 to December 1st, 2020. The recordings are encoded in



Opus format, with a 48kHz sampling rate. The dataset includes metadata such as age, gender, geolocation, previous respiratory conditions, and COVID-19 status, which is categorized into three classes: Healthy, COVID-19, and Symptomatic. The dataset has also been annotated by physician experts for various items including cough quality, dyspnea, wheezing, stridor, choking, and nasal congestion. Further details on the dataset will be presented in the next sections as we discuss the preprocessing steps.

## 4.2.2 Data cleaning and preparation

### 4.2.2.1 No COVID-19 status omission

It is important to note that COUGHVID dataset includes data with both known (Healthy, Symptomatic, and COVID-19) and unknown COVID-19 status. However, since our study is focusing on a supervised task, we have excluded all samples without COVID-19 status (i.e., no status provided) from our analysis. This resulted in the removal of 11326 samples from the dataset. The new total number of recordings becomes 16224.

### 4.2.2.2 Data formatting and silence removal

In this step, first, we converted the dataset recordings' webm format to wav format to fulfill our software requirements. We used Python library *soundfile* for this conversion. The conversion process was carried out by first reading each webm file using *Librosa* library and then writing the audio file to wav format using *soundfile* library. Wav files have been downsampled to 22 kHz to reduce complexity. This step resulted in the creation of a new directory containing the wav files with the same filename as the corresponding webm files. Next, in the silence removal step, we aimed to remove the silence regions from the recordings to ensure that our model would not learn from silence, and only learn from the informative regions. To achieve this, we used the Python library *Unsilence*, which is an open-source library for detecting and removing silence regions from audio files. The library works by first detecting the silent regions using a set of thresholds and then removing them by interpolating the signal in the region. We applied the library to all the recordings in the dataset, resulting in new files that only contained the informative regions of the cough sounds. We used the default parameters of *Unsilence* were a minimum duration of the silence of 0.25 seconds, a threshold of -60 dB, and a minimum non-silent length of 0.1 seconds. During the silence removal step, we found that 142 recordings in the COUGHVID dataset had zero duration after removing the silent regions. These recordings were considered as empty and were removed from the dataset. Table 4.1 shows the new class label distribution after applying the silence removal step with the total duration in hours, and Figure 4.2 shows an example of the removed regions from an audio recording.

Table 4.1: COVID-19 status label distribution over the three statuses *Healthy*, *COVID-19*, and *Symptomatic* after removing *No-COVID-19 status* and *empty signals* during silence removal step

	Healthy	Symptomatic	COVID-19	Total
Samples	12377	2567	1138	<b>16082</b>
Duration (h)	14.26	3.01	1.44	<b>18.71</b>

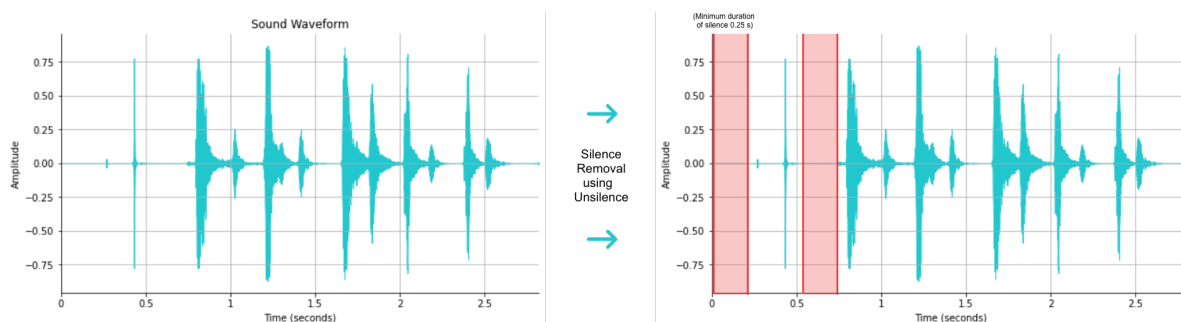


Figure 4.2: Example of the applied silence removal from an audio signal

After applying the silence removal step to the audio samples, we took a small set of samples and manually listened to them to check if any important hearable patterns were preserved. This was done to ensure that the silence removal step did not remove any important audio signals. Out of the 100 samples that were manually checked, we did not find any instances where the silence removal step removed any important audio signals.

#### 4.2.2.3 Cough detection analysis

Each recording has a corresponding *cough\_detected* entry in the metadata, the original study [91] provides the probability that the sound captured in the recording is a cough, under this entry. XGBoost was trained on 68 Prosodic, Spectral, and Cepstral features while a nested Cross-Validation procedure has been used to evaluate a cough classification model based on these audio features. The dataset has been partitioned into 10 equal segments, known as the "outer CV folds", and performed a separate Cross-Validation procedure on each fold to optimize the hyperparameters of the XGBoost model. The optimized XGBoost model was then tested on the testing data of its respective outer fold. In the model development, 22% of the recordings were randomly selected for assessing the generalization capabilities of their model to unseen test participants where the remaining 78% of the data was used to train the final XGBoost classifier and determine a final set of hyperparameters using a 5-fold random permutation-based ShuffleSplit Cross-Validation. According to their report, samples confirmed by experts to contain

coughs have a *cough\_detected* value of 1.0, while samples noted by experts to not contain coughs have a *cough\_detected* value of 0. Finally, the performance of the model has averaged across the 10 outer Cross-Validation folds. XGBoost model achieved a mean ROC-AUC of 96.4% and a standard deviation of only 3.3%, indicating that the model performed well and showed little variance among different testing groups, which yielded to recommending the threshold value of 0.8 to decide whether a recording is containing a cough or non-cough for COUGHVID dataset users. However, we wanted to investigate the impact of choosing different cough detection threshold probabilities on the performance of our model. We thus tested our model’s performance using different threshold probabilities ranging from 0.6 to 0.9. Through our extensive experiments, we found that setting the cough detection threshold probability to 0.7 resulted in the best performance. Table 4.2 shows the new class distribution after omitting all recordings which have a *cough\_detected* probability threshold under 0.7.

Table 4.2: COVID-19 status label distribution over the three statuses *Healthy*, *COVID-19*, and *Symptomatic* after omitting the recordings with *cough\_detected* value under 0.7

	Healthy	Symptomatic	COVID-19	<b>Total</b>
Samples	8958 (77.06%)	1935 (16.65%)	731 (6.29%)	<b>11624 (100%)</b>
Duration (h)	9.92	2.10	0,9082	<b>12.92</b>

#### 4.2.2.4 Conversion to binary classification problem

One of the key challenges in using COUGHVID dataset for COVID-19 detection is the significant class imbalance which is shown in Table 4.2, with only 731 positive COVID-19 cases compared to 8958 non-COVID-19 cases (6.29% of the dataset). To address this issue, we opted to combine the positive COVID-19 cases with the symptomatic cases under a single class, namely *Likely-COVID-19*. This decision was based on the understanding that symptomatic patients with a cough could potentially have COVID-19, and grouping them together under the same class could help boost the number of positive cases in the dataset. The remaining cases were labeled as *Non-Likely-COVID-19* or healthy cases. This way, the multi-class classification scheme was converted into a binary classification scheme. This approach has been adopted as the initial step to address the dataset’s skew, and further issue handling will be performed by the upcoming data augmentation pipeline detailed in Section 4.2.5. Table 4.3 shows the new class labeling with their sample count and the total number of hours.

Table 4.3: The new classes namely *Likely-COVID-19* and *Non-Likely-COVID-19* after combining *Positive COVID-19* and *Symptomatic* classes

	Non-Likely-COVID-19	Likely-COVID-19	Total
Samples	8958 (77.06%)	2666 (22.94%)	<b>11624 (100%)</b>
Duration (h)	9.92	3.0	<b>12.92</b>

### 4.2.3 Mel-Spectrogram computation

Audio time-frequency representation has shown an effective way to extract discriminative features from audio signals using DL architectures. One of the most common representations is the Mel-spectrogram (see Section 2.7.2.1) which has shown promising performance using DL approaches [8, 29, 32, 55]. It converts the time-domain signal into a frequency-domain representation, where the frequency spectrum is divided into equally spaced frequency bins according to the Mel scale. Due to its ability to capture important acoustic features of the signal, such as formants, harmonics, and timbre, in this study, we adopted the mel-spectrogram as the input for our DL models to capture the most salient cough-related features. Computing Mel-spectrograms involves several parameters, including window size  $n\_fft$ ,  $hop\_length$ , and the number of Mel-frequency bins  $n\_mels$ . These parameters have a significant impact on the resulting mel-spectrograms and can significantly affect the performance of any subsequent ML or DL models. In order to carefully choose the best parameters, we made use of GridSearch which yielded the best set of parameters. The details of the implemented setup and GridSearch will be detailed in the upcoming sections. We show in Figure 4.3 original audio signals with their corresponding computed Mel-spectrogram for two *Likely-COVID-19* cases and two *Non-Likely-COVID-19* cases, where several observations were made. The mel-spectrograms for both healthy cases (1 and 2) showed a similar magnitude distribution, with regions H11, H12, H13, H14, H15, and H16 displaying a similar concentration of magnitudes from 0 Hz to 2048 Hz. The same trend was observed for regions H21, H22, and H23. Additionally, the cough segments in the healthy cases shared a consistent magnitude distribution for each 0.5-second period. On the other hand, the COVID-19 cases (1 and 2) showed a different magnitude distribution in their mel-spectrograms. Regions C11, C12, and C13, as well as regions C21 and C22, showed a concentration of magnitudes below 1024 Hz. Moreover, it was observed that the magnitude of cough segments in *Likely-COVID-19* cases did not follow a consistent distribution over time, as seen in the *Non-Likely-COVID-19* mel-spectrograms. Specifically, in regions C11 and C21, it can be noted that the magnitude decreased over time, unlike *Non-Likely-COVID-19* cases where the selected regions shared almost the same magnitude distribution. Overall, by observing the magnitudes concentration in specific regions of the mel-spectrograms, we shall note that the latter may carry on several hidden and complex features for both *Non-*

*Likely-COVID-19* and *Likely-COVID-19* classes, which can be detectable by our DL model. It is also important to note that the identified regions of interest in the Mel-spectrograms may be influenced by non-cough or background noise patterns, and it is possible that other patterns relevant to the classification task may exist as well.

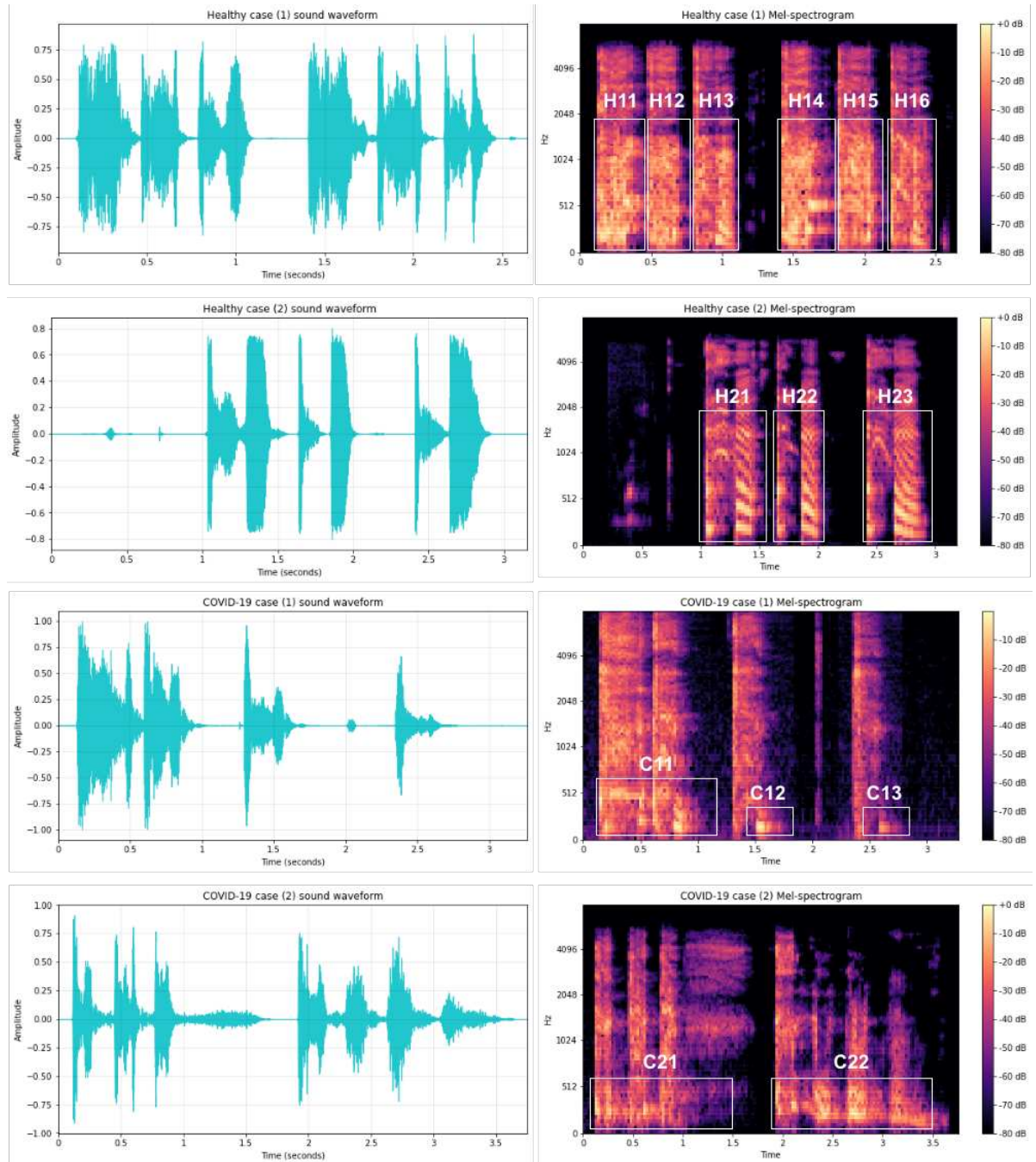


Figure 4.3: Audio waveform and corresponding mel-spectrogram for two *Non-Likely-COVID-19* cases (1 and 2) and two *Likely-COVID-19* cases (1 and 2), with each region of interest labeled H11, H12, H13, H14, H15, and H16 for *Non-Likely-COVID-19* case (1); H21, H22, and H23 for *Non-Likely-COVID-19* case (2); C11, C12, and C13 for *Likely-COVID-19* case (1); and C21 and C22 for *Likely-COVID-19* case (2). The x-axis represents time in seconds, while the y-axis represents frequency in Hz. The color scale indicates the magnitude in dB

#### 4.2.4 Input size standardization

In order to ensure that our DL models can achieve accurate and reliable results, it is important to standardize the inputs to a fixed size. This is because DL models require inputs of a fixed size, and if the inputs are not standardized, the models may produce inconsistent results. To achieve this standardization, we resized all recordings in our dataset to a standard length of 156027 (7.07 seconds), which was obtained by computing the mean length of all recordings. Samples with a duration longer than 7.07 seconds were truncated, while those with a duration shorter than 7.07 seconds were padded with zeros at the beginning and the end with equal amounts. By standardizing the inputs in this way, we can ensure that all recordings have the same length and format, which will allow us to train and evaluate our models more effectively. Figure 4.4 shows an illustration of zero-padding and truncating an audio signal with the corresponding mel-spectrogram.

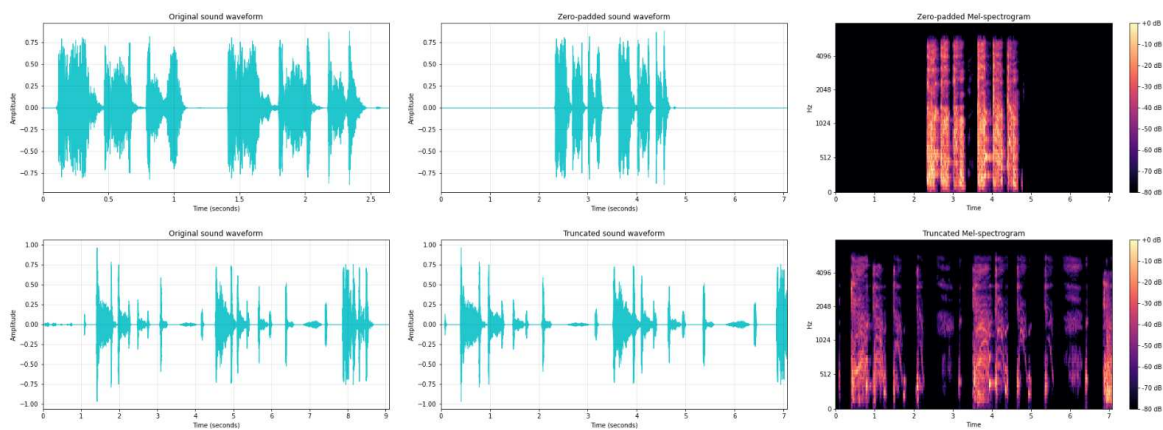


Figure 4.4: An example of two audio signals, the first with length 58275 (2.64s) which has been zero-padded to 7.07s, and the second with a length of 200407 (9.08s) which has been truncated to 7.07s

#### 4.2.5 Two-level data augmentation

As we have seen, COUGHVID dataset is imbalanced, even after combining *COVID-19* and *Symptomatic*. To address this issue, we proposed two levels of data augmentation to increase the size of the dataset and add more variability in terms of *Likely-COVID-19* cases. By applying these augmentation techniques, we can create additional cases and improve the generalization capability of our classification model. In the next subsections, we explore the two levels of data augmentation. The first level is performed directly on the original audio signal, and we utilized for that, pitch shifting, whereas the second involves using SpecAugment technique to augment the pre-computed mel-spectrogram.



#### 4.2.5.1 Signal-based data augmentation: Pitch shifting

Pitch shifting technique (see Section 2.9.2), has been widely used in the literature (see Table 3.3, 3.4, and 3.5), and has shown a significant impact in boosting ML and DL models (see Section 3.4.3). We utilized Librosa Python library for audio processing and analysis to implement pitch shifting. Specifically, we applied this method to the *Likely-COVID-19* class in order to increase the number of samples in the minority class. The augmentation involved shifting the pitch of audio samples down by four semitones ( $n\_step = -4$ ) to generate new samples.  $n\_step$  was carefully chosen through a manual review process where two independent listeners examined most of the augmented recordings to ensure that vocal features were not affected by pitch-shifting. By incorporating this level of data augmentation, we can add more variability to the dataset and increase the number of *Likely-COVID-19* cases. Figure 4.5 shows the pitch shifting of a *Likely-COVID-19* audio signal with the corresponding mel-spectrogram before and after applying pitch shifting. In addition to the manual review process of the augmented recordings, we highlight some regions of interest in both the original and pitch-shifted corresponding mel-spectrogram, in which, we can observe that almost all patterns are preserved in the pitch-shifted version.

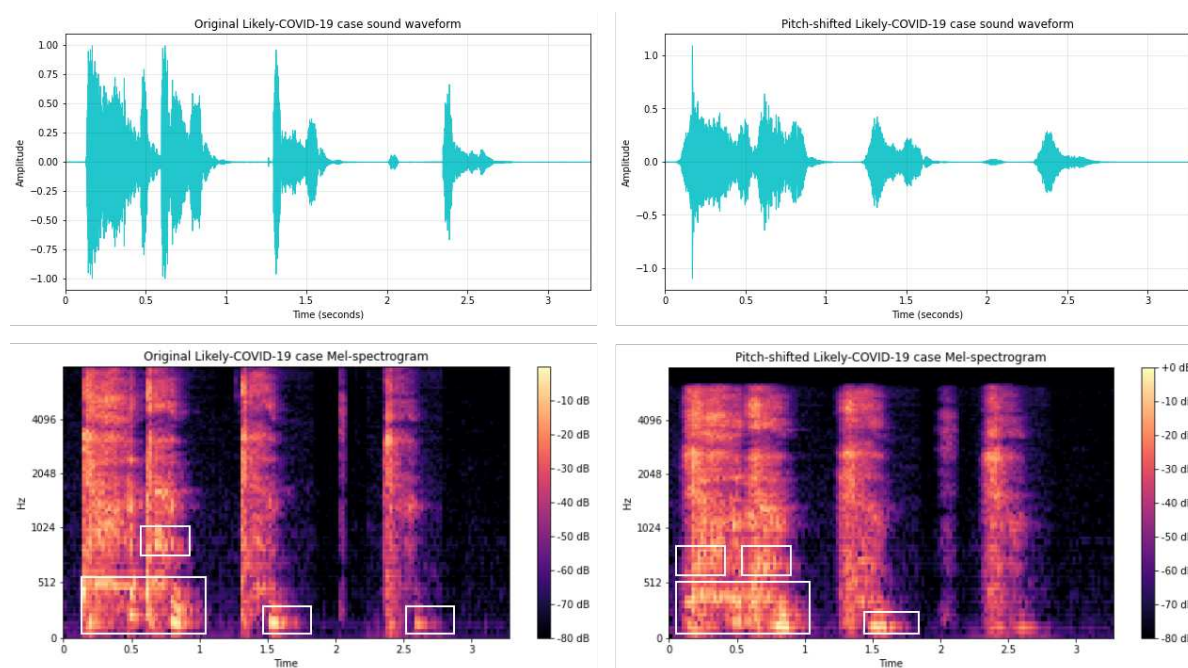


Figure 4.5: Utilizing pitch shifting technique with  $n\_step = -4$  on a *Likely-COVID-19* audio signal with the corresponding mel-spectrogram (original and augmented) with a highlight of the high magnitude regions.

#### 4.2.5.2 Spectral-based data augmentation: SpecAugment

The proposed SpecAugment technique is applied as the second level of data augmentation to increase the number of *Likely-COVID-19* samples in the dataset. This method was inspired



by the significant improvement of accuracy achieved by the application of SpecAugment in the DiCOVA challenge (see Section 3.4.3). SpecAugment consists of three-step augmentation method, the first step involves Time warping, where a random point along the horizontal axis passing through the center of the mel-spectrogram is warped to the left or right by a distance selected from a uniform distribution ranging from 0 to the time warp parameter  $W$ . In the second step, Frequency masking is employed to mask consecutive mel frequency channels  $[f_0, f_0 + f_r)$ , where  $f_r$  is selected from a uniform distribution ranging from 0 to the frequency mask parameter  $F$ , and where  $f_0$  is selected in  $[0, v - f_r)$  ( $v$  represents the number of mel frequency channels). Finally, Time masking is applied in the third step to mask successive time steps  $[t_0, t_0 + t_r)$ , where  $t_r$  is taken from a uniform distribution ranging from 0 to the time mask parameter  $T$ , and  $t_0$  is chosen from  $[0, \tau - t)$ , where  $\tau$  represents the number of timesteps of the mel-spectrogram. The mentioned steps are applied sequentially to obtain an augmented mel-spectrogram, however, they can be applied independently or in combination. In this study, we used a combination of Frequency masking and Time masking with masking parameters  $F = 30$  and  $T = 30$ , respectively. Generating more samples for the minority class (Likely-COVID-19) can help to improve the model’s ability to recognize and differentiate the positive cases from the healthy cases. With this respect, we used a combination of time and frequency masking to generate two new mel-spectrograms for the *Likely-COVID-19* class. By generating two samples, we aim to increase the variability and diversity of the augmented data for this class, which can help the model to learn more robust features and improve its generalization ability. On the other hand, we generated only one sample for the *Non-Likely-COVID-19* class to ensure diversity, also in the augmented data. The combination of time masking and frequency masking allows for a more diverse and effective augmentation of the mel-spectrograms. Time masking is useful for introducing variability in the time domain, allowing the model to learn more robust features and relationships in the time series. On the other hand, frequency masking helps to introduce variability in the frequency domain, which is particularly useful for handling variations in the spectral characteristics of the audio signals. By combining both time-masking and frequency masking, we can introduce more comprehensive and diverse forms of augmentation, thereby improving the overall performance of our DL model. Additionally, the use of both techniques helps in mitigating overfitting. Figures 4.6 and 4.7 show samples of the augmented mel-spectrograms for both classes using spectral data augmentation SpecAugment.

#### 4.2.6 Attention-based Hybrid CNN-LSTM Architecture

The attention-based hybrid CNN-LSTM architecture for COVID-19 diagnosis is presented in Figure 4.8. This architecture comprises four blocks. The first block adopts a CNN architecture that takes in augmented mel-spectrograms as input with a dimension of  $(39 \times 88 \times 3)$ . The convolution layers in this block extract the most informative and relevant features using different filters. In the second block, LSTM, the resulting feature maps from CNN block are forwarded to an LSTM

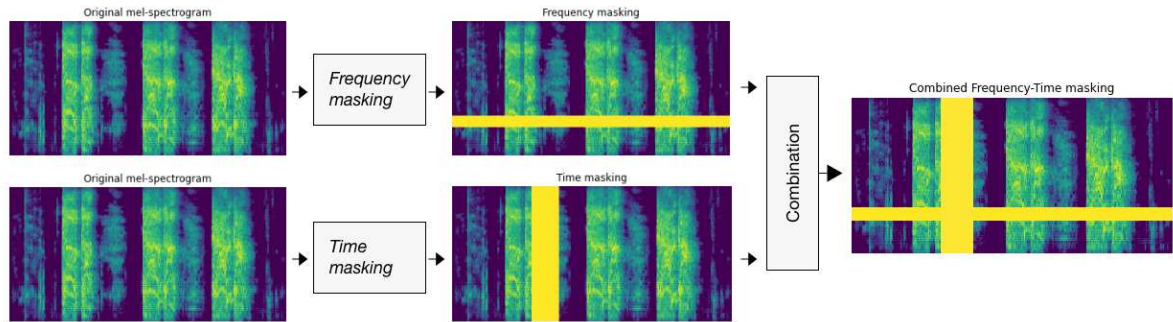


Figure 4.6: Illustration of SpecAugment for a Non-Likely-COVID-19 sample, a new mel-spectrogram is generated by combining *Frequency-masking* and *Time-masking*, for each mel-spectrogram in the dataset.

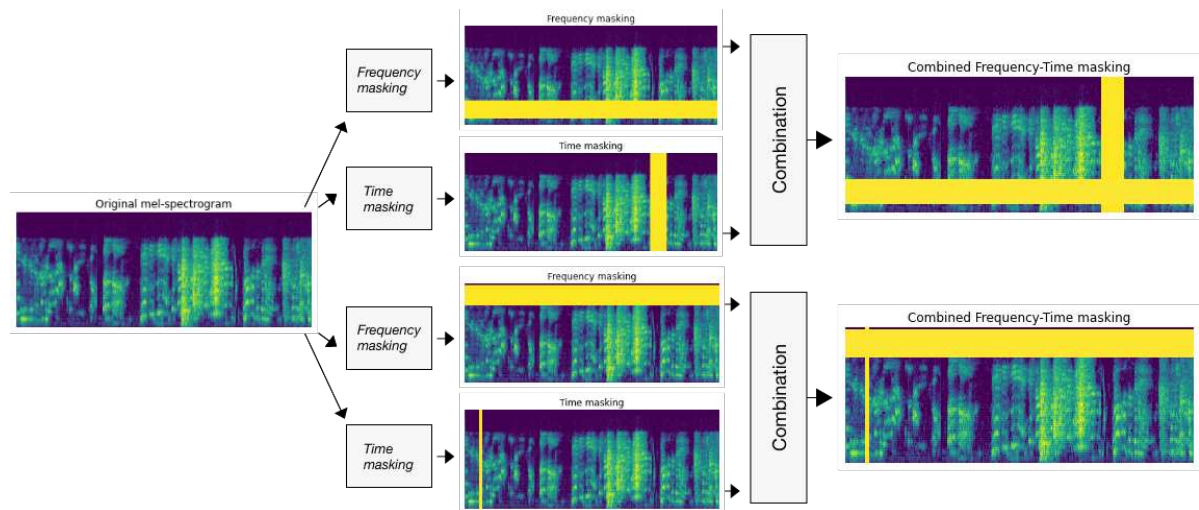


Figure 4.7: Illustration of SpecAugment for a Likely-COVID-19 sample, two new mel-spectrograms are generated by combining *Frequency-masking* and *Time-masking*, for each mel-spectrogram in the dataset.

layer. Here, deep features that have high temporal correlation are selected to be passed on to the third block which is the attention layer for capturing more useful patterns. The last block consists of a simple fully connected layer for feature learning and classification. The proposed network architecture is detailed in Table 4.4, in terms of layer type, parameters, and output size. The following subsections will describe the details of each block.

#### 4.2.6.1 CNN

The CNN block of our proposed architecture consists of four convolutional layers, each having 16, 32, 64, and 128 filters respectively, and a kernel size of  $(2 \times 2)$ . We follow each convolution layer with an Average Pooling layer, which is used to downsample the feature maps by linking them to a window of pre-fixed dimensions and a defined step unit called stride. Our pooling window has a

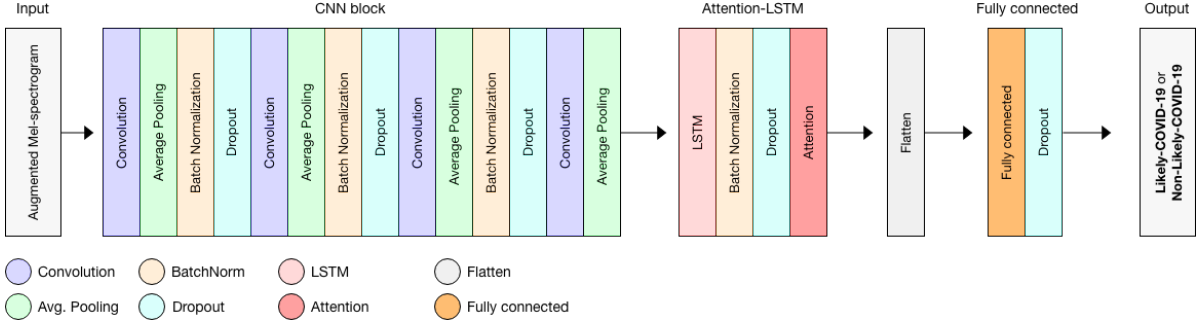


Figure 4.8: Block diagram of our proposed Attention-based hybrid CNN-LSTM architecture

size of  $(2 \times 2)$  and a stride of size  $(1 \times 1)$ . To increase the non-linearity of the feature maps, we use ReLU activation function  $f(x) = \max(0, x)$  which has been shown to be effective with CNNs. We further improve the model training process and speed up convergence by using BN to normalize the activations, and Dropout to avoid overfitting and improve generalization.

#### 4.2.6.2 LSTM

LSTM layer is a crucial component in our proposed architecture. It consists of three gates: input gate, forget gate, and output gate. The input gate decides which new information will be stored in the cell state, and the forget gate determines what information to discard from the previous cell state. The output gate decides what the final output will be. In our implementation, the input tensor has a shape of  $(31, 10240)$  which is the feature maps extracted by CNN block. Specifically, the output of the last pooling layer *AveragePool2D\_4* in the CNN block is reshaped after passing by BN and dropout, and then, fed as input to LSTM layer which processes the temporal sequence of feature maps by incorporating information from previous time steps and capture relevant patterns over time using the above-mentioned gates. One LSTM layer with 256 units has been used, which is activated by TanH, followed by BN and dropout layers to prevent overfitting.

#### 4.2.6.3 Incorporating additive attention mechanism

In order to improve the accuracy of our hybrid CNN-LSTM diagnosis model, we incorporated an additive attention mechanism which has been investigated in Section 2.5.5. Generally, attention mechanism is responsible for selectively focusing on the most important and informative parts of the decoder's output, LSTM hidden states, in our case. Our simple additive attention layer is composed of three phases, starting with the scores alignment phase where the hidden states  $H_t$  are passed through a trainable weight matrix defined by  $W_{att}$  and bias  $b_{att}$ , and then passed through a TanH activation function to compute the scores  $S_t$ . The next phase involves computing the attention weights  $\alpha_t$  using Softmax function. Finally, the context vector, or what we named *attention vector* is calculated as a weighted sum of all hidden states, where each weight  $\alpha_t$  is multiplied by its corresponding hidden state  $h_t$ . The output of the attention layer is then

Table 4.4: The proposed network architecture in terms of layer, output size. Each block (CNN, LSTM, Attention and Full connection) is surrounded by horizontal lines.

#	Layer	Details	Output size
1	Input	Mel-spectrogram (39×88×3)	-
2	Conv2D_1	16 filters of size (2×2), stride (1×1)	(38,87,16)
3	AveragePool2D_1	Window size (2×2), stride (1×1)	(37,86,16)
4	BN_1 + ReLU_1	-	(37,86,16)
5	Dropout_1	0.2	(37,86,16)
6	Conv2D_2	32 filters of size (2×2), stride (1×1)	(36,85,32)
7	AveragePool2D_2	Window size (2×2), stride (1×1)	(35,84,32)
8	BN_2 + ReLU_2	-	(35,84,32)
9	Dropout_2	0.2	(35,84,32)
10	Conv2D_3	64 filters of size (2×2), stride (1×1)	(34,83,64)
11	AveragePool2D_3	Window size (2×2), stride (1×1)	(33,82,64)
12	BN_3 + ReLU_3	-	(33,82,64)
13	Dropout_3	0.2	(33,82,64)
14	Conv2D_4	128 filters of size (2×2), stride (1×1)	(32,81,128)
15	AveragePool2D_4	Window size (2×2), stride (1×1)	(31,80,128)
16	BN_4 + ReLU_4	-	(31,80,128)
17	Dropout_4	0.2	(31,80,128)
18	Reshape_1	Reshape into recurrent layer input	(31,10240)
19	LSTM_1	256 units	(31,256)
20	TanH_5	-	(31,256)
21	BatchNorm_5	-	(31,256)
22	Dropout_5	0.2	(31,256)
23	Attention_1	-	256
24	Flatten_1	-	256
25	Dense_1	100 units	100
26	ReLU_6	-	100
27	Dropout_5	0.5	100
28	Dense_2	1 unit	1
29	Sigmoid_7	-	1
Total number of parameters			10,820,440
Number of trainable parameters			10,819,448
Number of non-trainable parameters			992

forwarded to a fully connected layer with 100 units for further feature learning, activated by ReLU function, and followed by dropout to prevent overfitting.

### 4.2.7 Baselines

In order to evaluate the performance of our proposed attention-based hybrid CNN-LSTM architecture, we compared it against three baseline models. The baselines were chosen to represent the main components of our proposed architecture: CNN, LSTM, and hybrid CNN-LSTM. By comparing the performance of our proposed model against these baselines, we aim to demonstrate the impact of each component on the overall performance of the model.

#### 4.2.7.1 CNN architecture

CNN baseline architecture is composed of two convolutional layers with 16 filters. Each one is followed by Average pooling layer and dropout layer to prevent overfitting. The model also includes a fully connected layer with 32 units for feature learning and classification.

#### 4.2.7.2 LSTM architecture

LSTM baseline model is composed of two LSTM layers, with the first layer having 128 units and the second layer having 256 units. The first layer is followed by BN to accelerate the model training process, while the second is followed by dropout to avoid overfitting. The model ends with a fully connected layer with 256 units that perform feature learning and classification.

#### 4.2.7.3 Hybrid CNN-LSTM architecture

Hybrid CNN-LSTM baseline is a combination of CNN and LSTM architectures. It includes four convolutional layers with increasing filter sizes of 16, 32, 64, and 128 respectively, followed by BN and dropout. After the convolutional block, the network includes a single LSTM layer with 256 units, also followed by BN and dropout. The output of the LSTM layer is then flattened and passed to a fully connected layer with 100 units, which is followed by a dropout layer to prevent overfitting. This architecture allows the model to learn both spatial and temporal features from the input data, while also preventing overfitting by using dropout and benefiting from BN.

## 4.3 Experimental Design

In all of our experiments, we utilized Adamax optimizer [65], which is a variant of Adam’s gradient descent algorithm that applies the maximum operation to the norm of the gradient. This approach results in more effective optimization in certain cases. The output layer of our model includes a single unit that is activated by Sigmoid function, producing a probability  $p$  of belonging to *Non-Likely-COVID-19* (class 0) or *Likely-COVID-19* (class 1), as shown in Equation 4.1. To compute the loss function, we used BCE, defined by Equation 4.2, which is applied to the scores generated by the Sigmoid activation function. Here,  $N$  denotes the total number of predicted data points,  $y_i$  represents the actual output, and  $\hat{y}_i$  is the predicted output.

$$(4.1) \quad y = \begin{cases} 0 & \text{if } p < 0.5 \\ 1 & \text{if } p \geq 0.5 \end{cases}$$

$$(4.2) \quad Loss = -\frac{1}{N} \sum_{i=1}^N (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

### 4.3.1 Hyper-parameters optimization

To optimize the performance of our deep learning architecture, we conducted a thorough hyper-parameter tuning stage using GridSearch approach (see Algorithm 2). Our search spanned over a variety of hyper-parameters including mel-spectrogram computation, the proposed data augmentation pipeline, and classification model parameters. Specifically, we explored different values for *hop\_length* (number of samples between consecutive frames), *n\_mels* (number of mel frequency bands or the height of spectrogram), and *n\_fft* (the size of the FFT, or window size) to improve the quality of the input mel-spectrograms. In addition, we experimented with different values of *n\_steps* for pitch shifting method and two different values of masking parameters (*T* and *F*) while applying SpecAugment for spectral data augmentation. Finally, we tested different values of batch size and learning rate for the classification model architecture. The results of our hyper-parameter tuning and selection process are summarized in Table 4.5. It is worth also noting that the selection of other hyper-parameters, such as the number of hidden layers, units, and dropout probability, was based on several rounds of manual fine-tuning processes.

Table 4.5: Mel-spectrogram computation, data augmentation and classification model architecture hyper-parameters tuning

	Tested values	Best value
<i>hop_length</i>	{128, 256, 512, 1024}	<b>512</b>
<i>n_mels</i>	{64, 128, 256}	<b>128</b>
<i>n_fft</i>	{512, 1024, 2048, 4096}	<b>512</b>
<i>n_steps</i>	{-1, -2, -3, -4, -5}	<b>-4</b>
F (Frequency masking parameter)	{30, 50}	<b>30</b>
T (Time masking parameter)	{30, 50}	<b>30</b>
<i>batch_size</i>	{64, 128, 256, 512, 1024}	<b>256</b>
<i>learning_rate</i>	{1e-05, 1e-04, 1e-03, 1e-02}	<b>1e-03</b>

### 4.3.2 Models training

In this study, all models, including both baselines and proposed model, were trained on Kaggle Notebook, which provides ample software and hardware resources for ML and DL tasks.

Specifically, Kaggle Notebook offers 43 hours of GPU usage and 20 hours of TPU usage per week, 19.6 GB of disk space, and 16 GB of memory. For better model performance validation, we used Cross-validation, which involves training the classifier on  $K - 1$  folds of data and using the remaining fold for validation and testing. This process was repeated  $K$  times, and the final result was obtained by averaging the  $K$  experiments. In our study, we set  $K = 10$ . Before starting the training process, we first split the whole dataset into two subsets: 80% for training and 20% for testing. Then, in each fold, 10% of the training set was used for validation, while the remaining 90% was used for training. Each model was trained for 500 epochs. By using this method, we ensured that the models were thoroughly trained and evaluated in a rigorous manner.

### 4.3.3 Performance evaluation

To evaluate the performance of our model and compare it with our baseline models, we computed six metrics: Accuracy, which measures the ratio of correct predictions over the total number of evaluated instances. Precision, which measures the ratio of correct predictions over the number of positive samples. Sensitivity, which measures the ratio of true positive predictions over the total number of positive samples. Specificity to measure the ratio of correctly classified negative cases over the total number of negative samples. F1-score that combines sensitivity and precision through harmonic mean, and finally, ROC-AUC which visualizes the tradeoff between true positive rate (TPR) and false positive rate (FPR). A higher TPR and lower FPR lead to a higher ROC-AUC score. In addition, it is important to use multiple metrics to ensure a comprehensive assessment of the model’s performance. Each metric provides a unique perspective on the model’s performance and limitations, and using multiple metrics can help identify areas for improvement. For instance, if only classification accuracy is reported, a model with high accuracy may have low sensitivity, indicating that it is correctly classifying negative samples but missing many positive ones. In this case, reporting the sensitivity is crucial to indicate the weakness of the model.

## 4.4 Results and findings

In this section, we present the results of our experiments for baselines and our proposed model. The metrics were computed by averaging the results obtained from 10-fold Cross-Validation, using the same test set for all models to ensure a fair comparison. We report the average (*avg*) and standard deviation (*sd*) of Cross-Validation  $avg \pm sd$  for each metric.

### 4.4.1 Baselines

LSTM baseline model, achieved an average accuracy of 76.41%, a sensitivity of 73.19%, and a ROC-AUC score of 76.26%, with a training time of two and a half hours. CNN baseline model, achieved an average accuracy of 83.37%, a sensitivity of 76.97%, and a ROC-AUC score of 83.00%, with a training time of three hours. Hybrid CNN-LSTM baseline model, which has the same

proposed architecture without additive attention layer, achieved the best performance with an average accuracy of 89.35%, an average sensitivity of 87.74%, and an average ROC-AUC of 89.28%, after 8 hours of training. Table 4.6 presents the Cross-Validation results of the three baselines, Figure 4.9 shows accuracy and loss curves for CNN, LSTM, and Hybrid CNN-LSTM baselines. 10-fold CV curves are plotted with discontinued lines, where the average is highlighted with a thick line.

Table 4.6: Results of 10-fold cross-validation for the proposed baselines (LSTM only, CNN only, and hybrid CNN-LSTM) using the aforementioned evaluation metrics.

	Accuracy	Sensitivity	Precision	Specificity	F1	AUC-score
LSTM	76.41±0.88	73.19±2.02	75.89±1.41	79.34±0.98	74.48±0.85	76.26±0.87
CNN	83.37±2.07	76.97±4.78	86.31±2.16	89.04±2.47	81.27±2.79	83.00±2.17
CNN-LSTM	<b>89.35±0.76</b>	<b>87.74±1.81</b>	<b>89.46±1.72</b>	<b>90.81±1.42</b>	<b>88.56±0.97</b>	<b>89.28±0.79</b>

The performance of the baselines on the test set was evaluated to validate the obtained results. The ROC curve for output probabilities is presented in Figure 4.10.

#### 4.4.2 Proposed approach

To show the performance and robustness of our approach, we provide the Cross-Validation details of our approach as well as the overall validation performance after 8 hours of training in Table 4.7. The curves illustrated in Figure 4.11 shows the development of our model during the training process in terms of accuracy, loss, sensitivity, and precision. A comparison between the proposed approach and the baselines reveals an overall improvement in all evaluation metrics, particularly in ROC-AUC and sensitivity scores. The complete comparison of the baselines with our developed approach is displayed in Table 4.8. Moreover, we exhibit the corresponding ROC curves for comparison, which clearly demonstrate the superiority of the proposed approach. We note that for a testing set of 6783 samples, the inference time was 2.84 seconds which means only 0.4 millisecond per instance.

### 4.5 Analysis and interpretation

This study aimed to develop a fast and effective method for diagnosing COVID-19 using solely cough sound recordings, with the goal of helping to limit the spread of the virus. The performance of the proposed framework is presented in the previous section. After pre-processing and filtering the original data by applying several steps including silence removal and multiclass classification to binary classification, a data augmentation pipeline that was applied to address the class imbalance problem, where there were 8958 negative samples versus only 2666 positive samples.



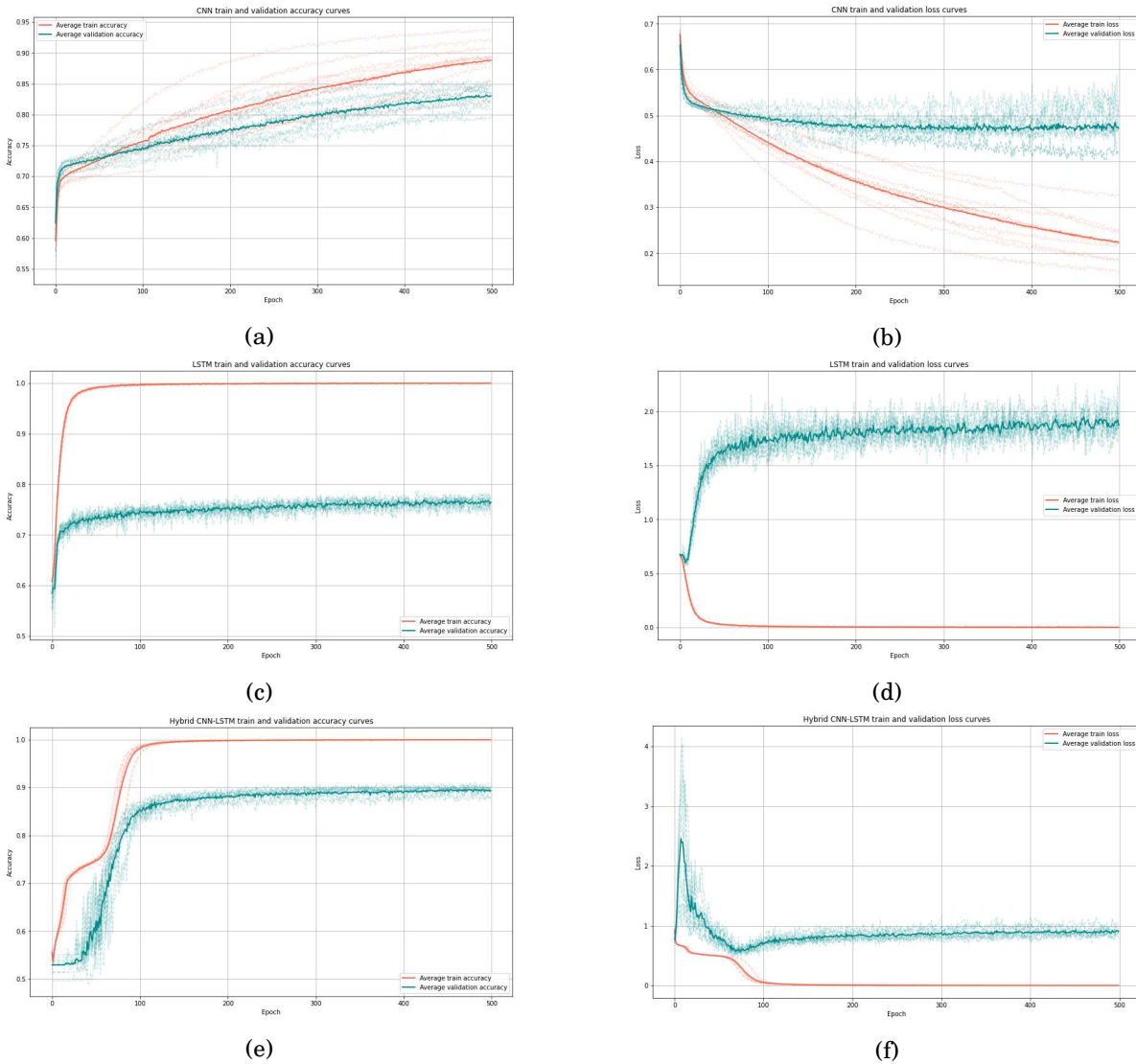


Figure 4.9: Accuracy and loss curves for baseline, (a) and (b) for accuracy and loss curves of CNN model, (c) and (d) for LSTM, where (e) and (f) for hybrid CNN-LSTM

The main components of the DL architecture used in this study were LSTM, CNN, and hybrid LSTM-CNN, which were used as baseline models for comparison. The results revealed that LSTM alone did not perform well, and show a notable presence of overfitting issue in Figure 4.9 (c) and (d) despite experimenting with different configurations, with an accuracy of only 77.75% and a sensitivity of 72.88%. This led to the conclusion that LSTM alone is unable to extract meaningful patterns from mel-spectrogram images and the model starts to overfit on the training data without performing well on testing data.

On the other hand, CNN performed better than LSTM in correctly classifying *Non-Likely-COVID-19* samples, achieving 89.73% correct predictions among all negative samples in the test set. Additionally, the sensitivity was improved from 72.88% achieved by LSTM to 81.53%, indicating

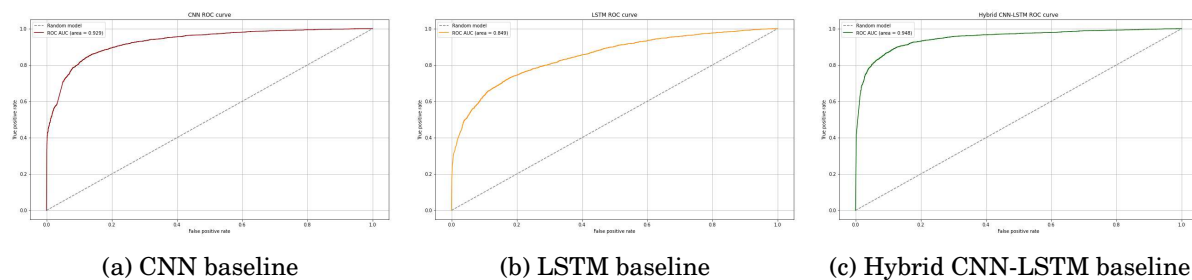


Figure 4.10: ROC curves of the three proposed baselines CNN, LSTM, and Hybrid CNN-LSTM which were computed from the model’s output probabilities

Table 4.7: The result of each fold of the 10-fold Cross-Validation for the novel approach (Attention-based Hybrid CNN-LSTM) in term of the above-mentioned evaluation metrics.

	Accuracy	Sensitivity	Precision	Specificity	F1	AUC-score
Fold-1	91.41	91.77	89.51	91.11	90.63	91.44
Fold-2	90.93	91.05	90.08	90.81	90.56	90.93
Fold-3	91.78	90.12	91.54	93.14	90.82	91.63
Fold-4	91.41	89.59	91.84	93.00	90.70	91.30
Fold-5	90.96	89.79	91.26	92.05	90.52	90.92
Fold-6	91.33	89.03	91.90	93.30	90.44	91.16
Fold-7	92.03	89.04	93.50	94.63	91.21	91.83
Fold-8	91.59	91.48	91.06	91.69	91.27	91.59
Fold-9	90.01	89.71	89.36	90.28	89.54	89.99
Fold-10	91.99	91.28	91.92	92.65	91.60	91.96
<b>avg±std</b>	<b>91.35±0.57</b>	<b>90.30±0.97</b>	<b>91.20±1.19</b>	<b>92.27±1.26</b>	<b>90.73±0.53</b>	<b>91.28±0.54</b>

that CNN has a greater ability to extract spectral and magnitude distribution-related features from mel-spectrogram images. The confusion matrices of both models are shown in Figure 4.13, where LSTM misclassified 866 out of 3,221 positive cases and 635 out of 3,562 negative cases (a), while CNN misclassified 595 positive samples and 366 negative cases (b).

To address the limitations of the baseline models, we thought about combining CNN-LSTM model to extract the most important spatial and spectral feature maps using the CNN block, and then passing the feature maps to the LSTM block where temporal correlations between features are extracted. This hybrid approach boosted the classification accuracy from 85.83% (achieved by CNN) to 88.44% and a sensitivity of 84.41%. From the confusion matrix in Figure 4.14 (a), we can see that the best error rate of 7.92% was achieved for negative cases where only 282 samples were misclassified. However, a total of 502 positive samples were misclassified.

As the need to build a diagnosis system with high sensitivity is crucial in medical cases, an attention mechanism module was added to the deep extracted temporal features to capture more informative patterns. This proposed approach achieved the best sensitivity rate of 90.93%

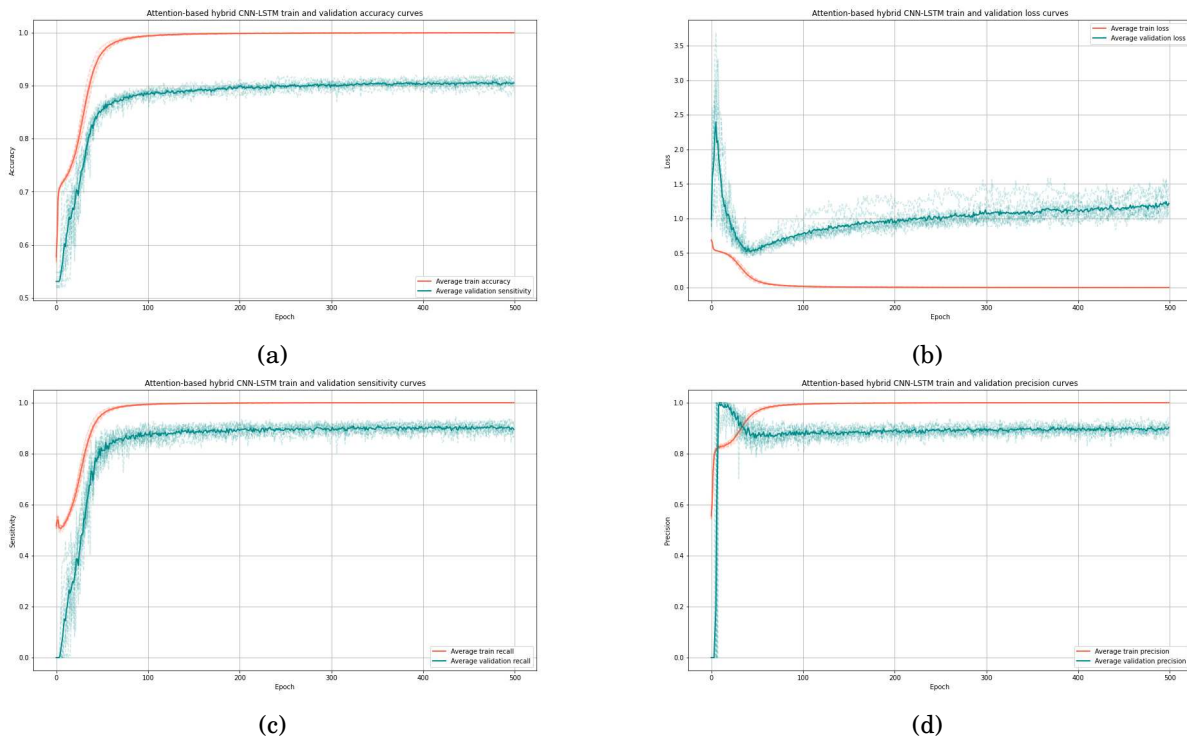


Figure 4.11: (a) Accuracy, (b) loss, (c) sensitivity, and (d) precision curves for our novel proposed approach Attention-based hybrid CNN-LSTM

Table 4.8: The performance in terms of the six evaluation metrics of our novel approach on the unseen data compared to the proposed baselines *A-CNN-LSTM* refers to Attention-based Hybrid CNN-LSTM

	Accuracy	Sensitivity	Precision	Specificity	F1	AUC-score
LSTM	77.75	72.88	78.76	82.17	75.71	77.52
CNN	85.83	81.53	87.81	89.73	84.55	85.63
CNN-LSTM	88.44	84.41	<b>90.64</b>	<b>92.09</b>	87.41	88.25
<b>A-CNN-LSTM</b>	<b>91.13</b>	<b>90.93</b>	90.47	91.31	<b>90.71</b>	<b>91.13</b>

compared to the best baseline (hybrid CNN-LSTM with 84.41%), a classification accuracy of 91.13% and F1-score of 90.71%. The best testing results in terms of precision and specificity (90.64% and 92.09%, respectively) were achieved by the hybrid CNN-LSTM without attention mechanism. The confusion matrix in Figure 4.14 (b) shows that only 292 positive cases were misclassified out of 3,221 positive testing samples.

The performance of the proposed approach with two-level data augmentation (Pitch shifting and SpecAugment) has been compared to the performance of the same model with only pitch shifting. Table 4.9 presents the comparison of the two scenarios, where results demonstrate that the proposed two-level data augmentation significantly improves the performance of the

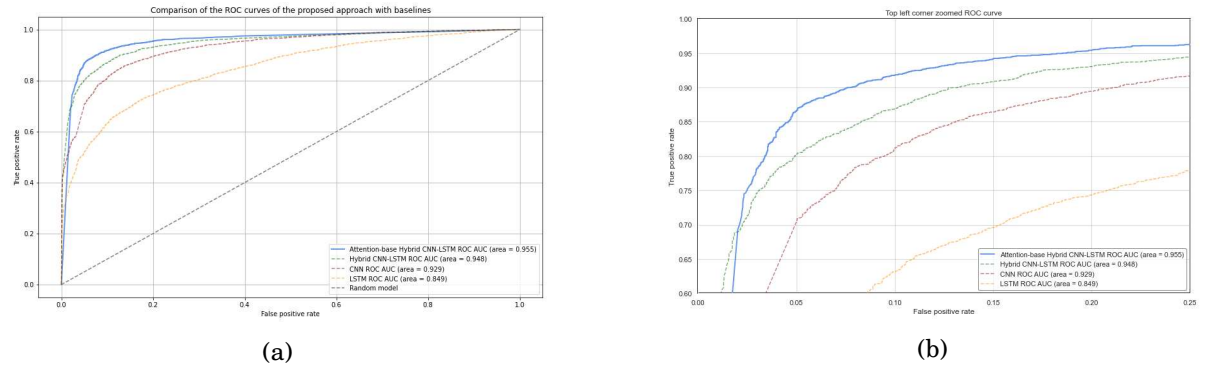


Figure 4.12: ROC curves of the baselines and our proposed approach. (a) displays the full ROC curves, while (b) zooms in on the top-left corner to highlight the differences.

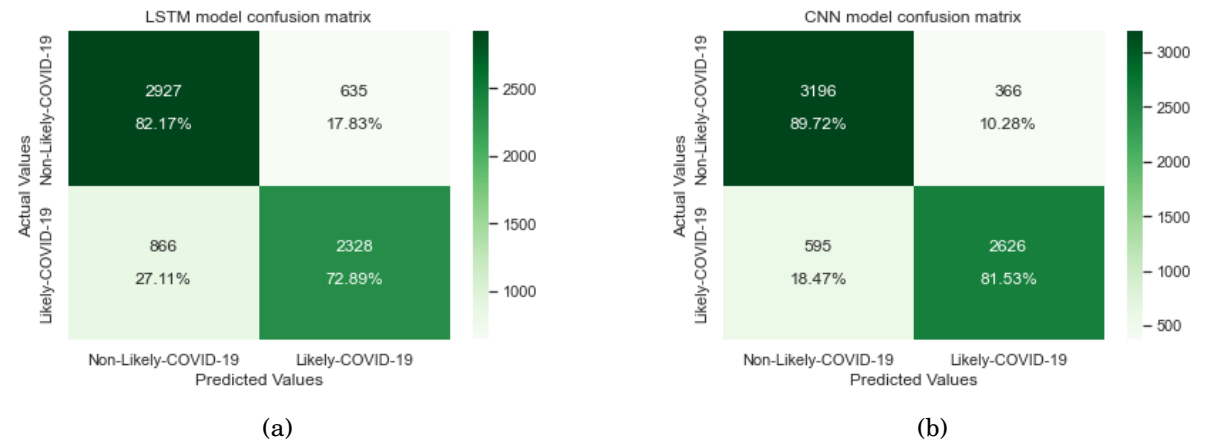


Figure 4.13: Confusion matrices for the test set: (a) LSTM model and (b) CNN model

model in all evaluation metrics. Specifically, we denote a very low sensitivity of 48.55% for pitch shifting only, which indicates that the model trained on only pitch shifting augmented data is biased to correctly classify *Non-Likely-COVID-19* with a specificity of 94.78% without focusing on the model’s sensitivity. Moreover, without data augmentation, the model was biased towards the positive class and achieved poor performance on the test set even with class weighting. This highlights the importance of the proposed two-level data augmentation in addressing class imbalance and improving the performance of the developed approach.

Table 4.9: The performance of our model with the proposed data augmentation (Pitch shifting (P-S) and SpecAugment) against pitch shifting only

	Accuracy	Sensitivity	Precision	Specificity	F1	AUC-score
P-S only	77.43	48.55	84.85	94.78	61.76	71.67
<b>P-S + SpecAugment</b>	<b>91.13</b>	<b>90.93</b>	<b>90.47</b>	<b>91.31</b>	<b>90.71</b>	<b>91.13</b>

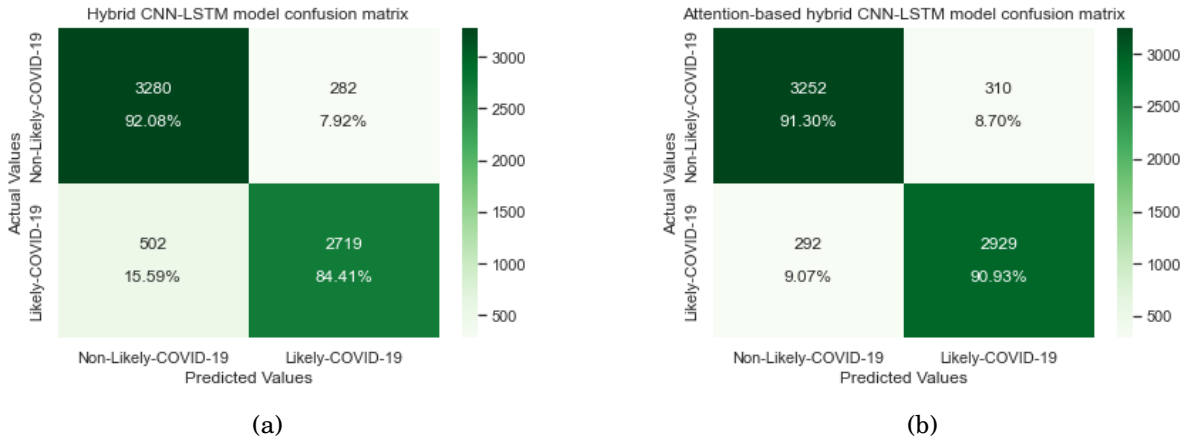


Figure 4.14: Confusion matrices for the test set: (a) Hybrid CNN-LSTM model and (b) Proposed Attention-based Hybrid CNN-LSTM model

In terms of overall performance, the best-performing classifier was the attention-based hybrid CNN-LSTM, which was able to distinguish between *Likely-COVID-19* and *Non-Likely-COVID-19* coughs with an ROC-AUC score of 91.13% (95.5% when computing with output probabilities). This comparison is illustrated in Figure 4.12 (b). Overall, these findings demonstrate the effectiveness of the proposed approach in accurately diagnosing COVID-19 using cough sound recordings and suggest the potential for this method to be used as a screening tool to help limit the spread of the virus.

## 4.6 Limitations of the approach

Despite the promising obtained results with quick and easy model deployment, our model is also prone to inherent weaknesses. In this section, we present the principal limitations of our work.

1. **Robustness of Cough Symptom for Discrimination:** The first limitation of the approach is related to the assumption that the cough training data is robust enough to differentiate between COVID-19 and non-COVID-19 cases. This assumption can be questioned as some characteristics of COVID-19 cough samples may also exist in patients with other diseases, as indicated in previous studies [128]. Moreover, the data preparation phase only considered healthy versus COVID-19 patients, leaving other potential gaps unexplored.
2. **Non-balanced dataset:** The class imbalance in the employed COUGHVID dataset represents a significant limitation to the proposed system. While the dataset contains more than 27,000 recordings, the number of positive cases is quite small compared to negative cases (1155 vs 12479). As a result, the performance of the proposed system is affected despite the employed data augmentation strategy.

3. **Impact of binary class transformation:** The third limitation of our proposed approach is related to the binary class transformation that was applied in this study. In order to simplify the classification task and increase the number of possible positive cases, we combined the Positive COVID-19 and Symptomatic classes into one single class. While this approach may reduce the false negative rate, it inevitably results in a loss of information and capacity to distinguish between COVID-19 and common symptoms. Therefore, the class labeling of Likely-COVID-19 and Non-Likely-COVID-19 may not fully reflect the complexity of the diagnostic process and may result in the misclassification of numerous cases.
4. **Performance and sensitivity:** Our proposed model achieved promising results in terms of correctly classifying unseen data with an accuracy of over 91%. The model also demonstrated a high level of sensitivity, surpassing previous studies [19, 84] that used different datasets. However, the error rate for classifying positive cases as negative is 9.07%, which is a critical type II error that demands further attention to deal with. Misclassification of positive cases as negative may result in unisolated patients, leading to the further spread of the virus.

## 4.7 Concluding notes

This chapter presents a COVID-19 diagnosis system based on cough sound recordings, utilizing a DL-based architecture for classification. Our contribution lies in the use of the largest publicly available cough dataset for COVID-19, COUGHVIG, and the implementation of two-level data augmentation techniques to address class imbalance and variability issues. Specifically, we applied signal data augmentation pitch shifting on positive class samples as in the first stage, followed by the spectral data augmentation technique SpecAugment, which involved randomly applying frequency and time masking on the mel-spectrograms. These approaches resulted in the generation of two new samples for the positive class and one for the negative class, thereby improving the performance of the model. We used hybrid CNN-LSTM and attention mechanism module, with single CNN, LSTM, and hybrid CNN-LSTM used as baselines to demonstrate the effectiveness of each part of our proposed architecture. Our best-performing model is the Attention-based hybrid CNN-LSTM, which achieved an overall classification accuracy of 91.13% and a sensitivity of 90.93%, outperforming the three baselines. The ability of our model to discriminate and distinguish between Likely-COVID-19 and Non-Likely-COVID-19 coughs is highlighted by a ROC-AUC score of 0.9113 on the unseen data. Our contributions to the field include the successful implementation of two-level data augmentation techniques to improve classification performance, as well as the use of a simple additive attention mechanism module to further performance enhancement. These advancements have important implications for the development of fast and effective COVID-19 diagnosis systems based on cough sound recordings, which have the potential to limit virus spread. In the next chapter, we will explore new approaches

for COVID-19 diagnosis from cough sound based on classical ML methods. We will investigate the use of a different feature type, class balancing technique, and classification method to further improve the performance of the diagnosis system. These approaches aim to build a simple and low-cost ML-based diagnosis system as well as addressing the same limitations and challenges, provide a comparative analysis of different approaches, and give a possible answer for our research question about the ability of ML-based models to efficiently diagnose COVID-19 from solely cough sound.





## MACHINE LEARNING APPROACHES BASED ON ENSEMBLE LEARNING FOR COVID-19 DIAGNOSIS FROM COUGH SOUND

### 5.1 Introduction

After a thorough investigation of a DL approach for COVID-19 diagnosis from cough sound in the previous chapter, we now shift our focus toward classical ML methods. One of our research questions is: *"Can ML methods achieve comparable performance to DL methods in developing accurate and efficient diagnosis systems for COVID-19 based on cough sound data?"* To answer this question, we explore the use of an ensemble learning method in combination with LLD, Autoencoders, and LLE. In the most of related works, ML approaches have shown great potential in diagnosing COVID-19 from cough sound. From the literature review, we observe that classical ML methods have been widely used and achieved acceptable and promising performance, specifically, in terms of sensitivity metric [19, 81, 128] even against some DL techniques [134]. In this chapter, we will present our works that are based on the use of an ensemble learning method for COVID-19 diagnosis from cough sound. Specifically, we will discuss three proposed approaches: (1) The use of Random Forest and original LLD vector [44], (2) Stacking Autoencoders for bottleneck-based feature extraction [137], and (3) Reducing LLD feature vector dimensionality using LLE [136]. Our works are all based on the use of an ensemble learning method, particularly Random Forest, and ComParE feature set. We were motivated to investigate the use of classical ML methods based on the promising results reported by previous works. This chapter is an extended version of the combination of our three works. The end of this chapter will show a comparison between the three proposed approaches in terms of detecting performance as well as time cost for training and inference.

## 5.2 Data exploratory and feature engineering

### 5.2.1 Data preparation

In this chapter, the same data preparation process as the previous chapter has been followed and applied to COUGHVID dataset. This includes the elimination of recordings without COVID-19 status, formatting the data, removing silence, and merging the classes *COVID-19* and *Symptomatic* into one single class to turn the problem into a binary classification problem. Additionally, we did not filter recordings by *cough\_detected* probability. This means that all recordings were included in the training process, regardless of the probability of detecting a cough in order to help increase the generalization of our models. Table 5.1 shows overall statistics about COUGHVID dataset after applying the aforementioned steps.

Table 5.1: Overall statistics about COUGHVID dataset at this stage (No-status elimination, Silence removal, and Multi-class to binary classification problem). Total length in *hours*, where maximum, minimum, and mean in *seconds*.

	Likely-COVID-19	Non-Likely-COVID-19
Samples count	3705	12377
Percentage	<b>23.03%</b>	<b>76.97%</b>
Total length	4.45	14.26
Minimum	0.51	0.49
Maximum	10.05	14.15
Mean $\pm$ sd	4.33 $\pm$ 2.19	4.14 $\pm$ 2.12

### 5.2.2 LLD ComParE 2016 feature set extraction

LLDs are a set of features that are used in audio signal processing, they capture basic properties of the audio signal that can be relevant and useful for many applications such as speech recognition, and audio classification tasks, in general. In simple words, LLDs refer to the set of features that are extracted in a preprocessing step for further audio, or speech analysis task, such as spectral roll-of, loudness, and temporal patterns. After providing an overview of LLD, the focus of the rest of this section will be on the **ComParE**, specifically, *ComParE\_2016* feature set, which is considered the largest and most commonly used LLD feature set. It refers to the Interspeech 2016 Computational Paralinguistics Challenge [107]. However, this feature set has been introduced and used in earlier works in the same challenge [106, 142]. Compared to other smaller variants *GeMAPSv01a*, *GeMAPSv01b*, and *eGeMAPSv01a*, *eGeMAPSv01b*, *eGeMAPSv02* which are available under the open source Python library *openSMILE*<sup>1</sup>, these variants vary from 18 to 88 features, *ComParE\_2016* is considered the largest LLD feature set as it includes 6373 low-

<sup>1</sup><https://github.com/audeering/opensmile>

level features, they have been divided into three main groups: *Energy-related LLD* (4), *Spectral-related LLD* (55), and *Voicing-related LLD* (6). Moreover, *ComParE\_2016* feature set includes statistical functions that are applied to the LLD,  $\Delta$ LLD (velocity), and  $\Delta\Delta$ LLD (acceleration) features, such as mean, standard deviation, linear regression slope, quadratic error, quadratic regression a, percentile range from 1% to 99%, as well as 6% and 94% percentiles. These statistical functions aim to extract more hidden patterns and statistically, explore the feature distribution, yielding a total of **6373** features. Table 5.2 presents the list of the mentioned LLD. In our study, we extracted this large feature set from the pre-processed cough recordings using OpenSMILE library, which is a widely used tool for audio feature extraction. *ComParE\_2016* feature has been investigated in the literature for COVID-19 diagnosis from cough sound like [117] in DiCOVA challenge and other related tasks [116], and it has shown acceptable performance.

Table 5.2: ComParE feature set [106, 107, 142]

Group	Features
Energy related LLD (4)	<ul style="list-style-type: none"> <li>- Sum of auditory spectrum (loudness)</li> <li>- Sum of RASTA-style filtered auditory spectrum</li> <li>- Zero-Crossing Rate</li> <li>- RMS Energy</li> <li>- F0 (SHS and viterbi smoothing)</li> </ul>
Spectral-related LLD (55)	<ul style="list-style-type: none"> <li>- RASTA-style auditory spectrum, bands 1–26 (0–8 kHz)</li> <li>- Spectral energy 250–650 Hz, 1 k–4 kHz</li> <li>- Spectral roll off point 0.25, 0.50, 0.75, 0.90</li> <li>- Spectral flux, centroid, entropy, slope</li> <li>- Psychoacoustic sharpness, harmonicity</li> <li>- Spectral variance, skewness, kurtosis</li> </ul>
Voicing-related LLD (6)	<ul style="list-style-type: none"> <li>- Probability of voicing</li> <li>- Log. HNR, Jitter (local, delta), Shimmer (local)</li> </ul>

### 5.2.3 Exploratory Data Analysis

This analysis is an essential step to understand the dataset and the characteristics of the features that can contribute to build an efficient classifier. In this study, we performed exploratory data analysis to gain insights into the distribution of the LLD features extracted from COUGHVID dataset recordings. Figure 5.1 presents box and whisker plot, showing the distribution of the most six important features extracted using RFE with Random Forest estimator. These features include RMSE, MFCCs, and Fundamental Frequency (F0). The plot indicates that there is not much difference in the distribution of these features between the two classes. The boxes for both classes are typically the same, with some outliers indicating a slight deviation from the typical distribution. This suggests that these features may not be significant indicators of COVID-19

status, and other features or combinations of features may be needed to accurately classify COVID-19 cases.

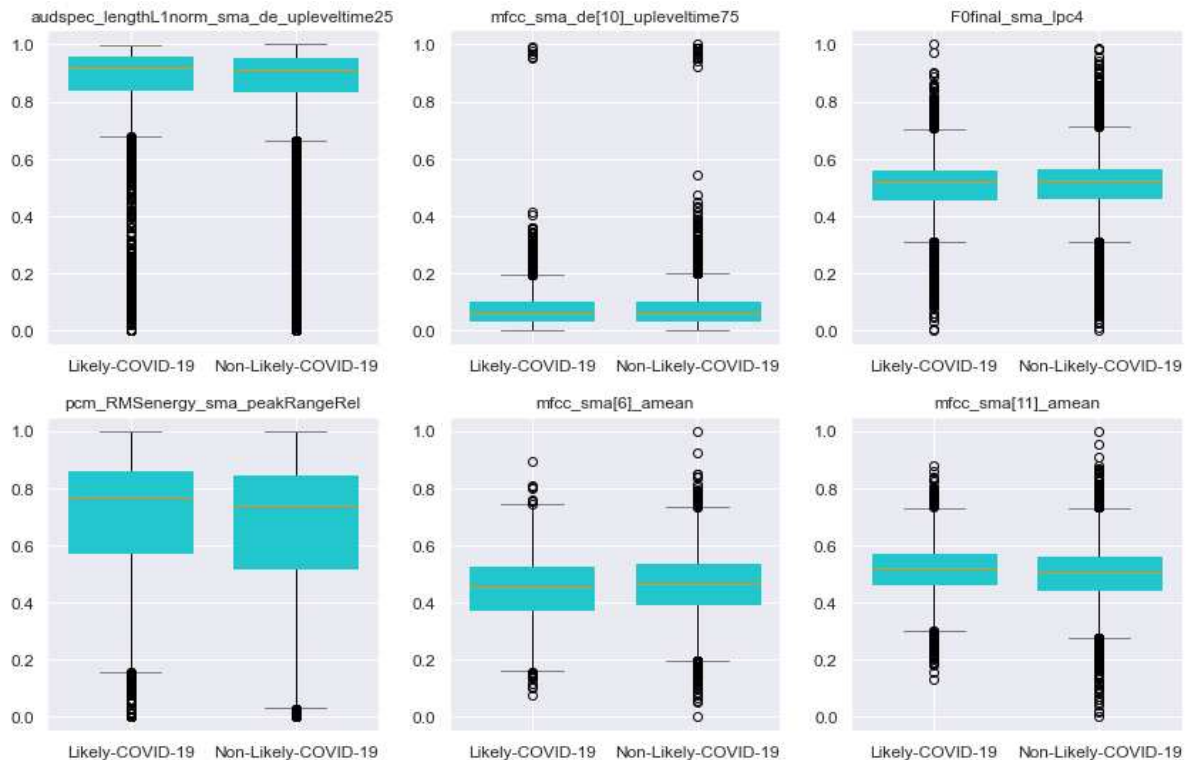


Figure 5.1: Box and whisker plot of the six most important features extracted using RFE. No significant differences between the two classes are observed, and multiple outliers are present. This suggests the potential need to further data engineering investigation in the dataset to find a discrimination between the two classes.

The presented box and whisker plot highlighted the need for further data exploration to better understand the distribution of the data and the potential discriminative power of the extracted features. This led us to perform a scatterplot matrix of the first three embeddings, illustrated in Figure 5.2, which were obtained through LLE. The diagonal of the matrix shows the density distribution of each embedding, we used Kernel Density Estimation (KDE), which facilitates a more accurate depiction of the data by providing a smooth and continuous estimation of the distribution. Despite the observed class overlapping, we observe also a trend of high density, specifically for Non-Likely-COVID-19 class, which can present a possible class separability, thus, the selected embeddings may able to capture meaningful differences between classes. This observation supports the hypothesis that training our classifier on dimension-reduced data can be more beneficial and can achieve better performance, as well as highlighting the importance of dimensionality reduction in the effective feature selection, and eliminating redundancy. Hence, by evaluating the performance achieved by our proposed approaches; bottleneck-feature-based approach and LLE dimensionality reduction, we can obtain a comprehensive confirmation of our

hypothesis.

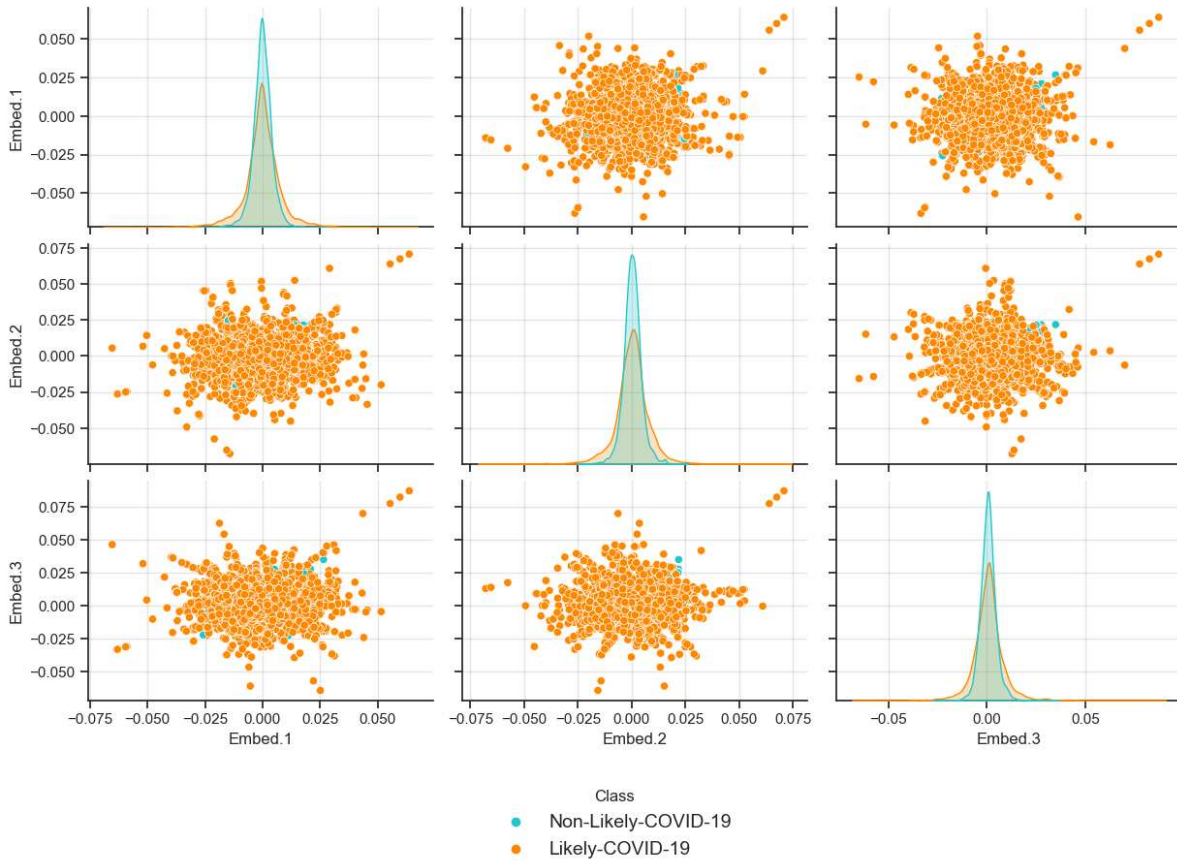


Figure 5.2: Scatterplot matrix of the first three embeddings of LLE where diagonal shows the distribution of embeddings values using KDE, illustrating a trend of high density for Non-Likely-COVID-19 class

#### 5.2.4 Class balancing by oversampling: SMOTE

As demonstrated in the previous chapter, our findings reveal that the model trained on an imbalanced dataset exhibits low performance and bias towards the majority class. Similarly, as the present work is based on the same dataset, we are still facing a class imbalance issue and we need to address it before proceeding with the training of our classifier. We employ SMOTE to balance the dataset by oversampling the minority class, by creating equality among the classes. This approach has been used in the literature and demonstrated to be effective in similar scenarios [25, 92]. SMOTE method begins by randomly selecting a minority class instance  $B$  and searching for its  $k$ -nearest minority class neighbors. Then, a synthetic instance is created by picking one of the  $k$ -nearest neighbors  $A$  at random and connecting  $A$  and  $B$  in the feature space to construct a line segment. The new synthetic samples are made by convexly joining the two chosen cases  $A$  and  $B$ , which leads to the generation of new synthetic minority samples

where each one is defined by  $N_{gen} = A + (A - B) * \lambda$  where  $\lambda$  is a random number between 0 and 1 (see Section 2.9.1 and Algorithm 5). Figure 5.3 presents scatterplots of the first two principal components (PC1 and PC2) of the data, one for the dataset before oversampling using SMOTE (a), and the other after oversampling (b). The aim is to demonstrate that the distribution of the data remains the same even after oversampling the minority class. The scatterplots are color-coded according to the two classes, and we can observe that the color distributions are similar in both plots. This suggests that the oversampling technique used in SMOTE has successfully created synthetic samples that maintain the distribution of the original data, which can address the class imbalance issue.

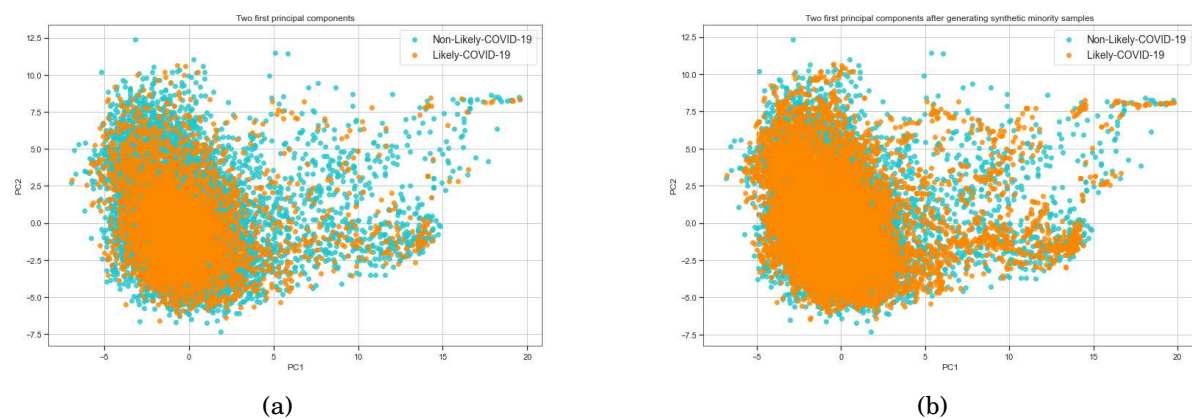


Figure 5.3: Scatterplots of PC1 and PC2 before (a) and after (b) SMOTE to show similar data distribution after oversampling the minority class

To further assess the effectiveness of SMOTE, we calculated the explained variance ratio for the first ten principal components before and after oversampling. As shown in the scree plot in Figure 5.4, there is a small difference in the explained variance ratio between the two cases (0.11 vs. 0.12 for the first component PC1). Although the difference is not significant, it suggests that SMOTE has addressed the class imbalance issue while not significantly altering the original data structure.

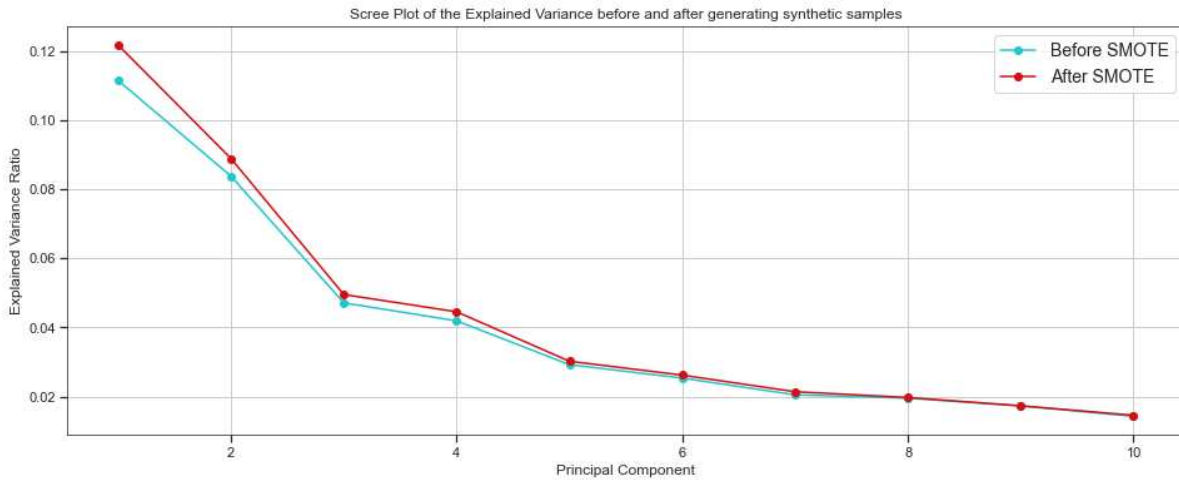


Figure 5.4: Scree plot showing the explained variance ratio of the top 10 principal components before and after oversampling the minority class using SMOTE

After providing a summary of data analysis and addressing the class imbalance issue, we move on to the next stage of our analysis, where we explore different feature extraction and dimensionality reduction techniques. Specifically, we consider three different approaches: first, forwarding the full-length LLD feature vector to the classification model, extracting bottleneck-based features using a stacked autoencoder, and reducing dimensionality using LLE. Random Forest (see Section 2.4.4 and Algorithm 3) will be used in the three approaches. We focus on the sensitivity metric, which is of utmost importance in our context, and compare the performance of these approaches against each other as well as other ML-based algorithms.

### 5.3 Approach evaluation and validation

In order to ensure the reliability and generalizability of our models, we used 5-fold Cross-Validation as a validation technique. This approach involves splitting the dataset into five equal parts, where each part is used once for validation and the other four parts are used for training. This process is repeated five times, ensuring that each part of the dataset is used once for validation. To evaluate the performance of our models, the same evaluation metrics as the previous chapter were used; accuracy, precision, sensitivity, F1-score, and specificity. These metrics provide a comprehensive assessment of the model’s performance by measuring different aspects of its classification ability, especially the ability to correctly identify true positives (Likely-COVID-19 cases), which is considered the most important metric in our context. We shall note that training and evaluating the three approaches have been done on a local machine, MacBook Pro, MacOS 11.7.4, 2,3 GHz Quad-Core Intel Core i7, and 16 GB of memory, thus, all time-cost performance have been computed based on the capacity of this hardware configuration.

## 5.4 Full-length LLD feature vector

In this section, we present the model of the first approach which utilizes the full-length LLD feature vector without any feature selection or dimensionality reduction technique. This approach was named LLD-RF.

### 5.4.1 Methodology description and model training

The first approach consists of forwarding the full feature vector of size 6373 to a Random Forest classifier. The model is as well, trained on COUGHVID dataset, which was preprocessed to extract LLD features using OpenSMILE Python library. The preprocessing steps included data formatting, resampling to 22kHz, silence removal, and finally, the class balancing with SMOTE to address class imbalance issue. We then, trained our classifier using 5-fold Cross-Validation after utilizing GridSearch to tune the hyperparameters of the classifier, where we focused on: the number of trees ( $n\_estimators$ ), the use of the whole training set to build each tree (*bootstrapping*), and the maximum number of features ( $max\_features$ ) to consider while conducting the best split. We based on the default values provided by *Scikit-learn*<sup>2</sup> of the rest of the parameters. Table 5.3 shows the tuned parameters and the selected values and Figure 5.5 presents a general overview of the approach. The dataset was split into two parts: 80% for training and 20% for testing.

Table 5.3: Tuned hyper-paramaters for the first approach LLD-RF.  $\sqrt{F}$  presents the square root of  $F$  which is the total number of features (6373)

	Tested values	Best value
$n\_estimators$	[200,1200]	<b>200</b>
<i>bootstrapping</i>	{Used, Not used}	<b>Not used</b>
$max\_features$	{ $F$ , $\sqrt{F}$ }	$\sqrt{F}$

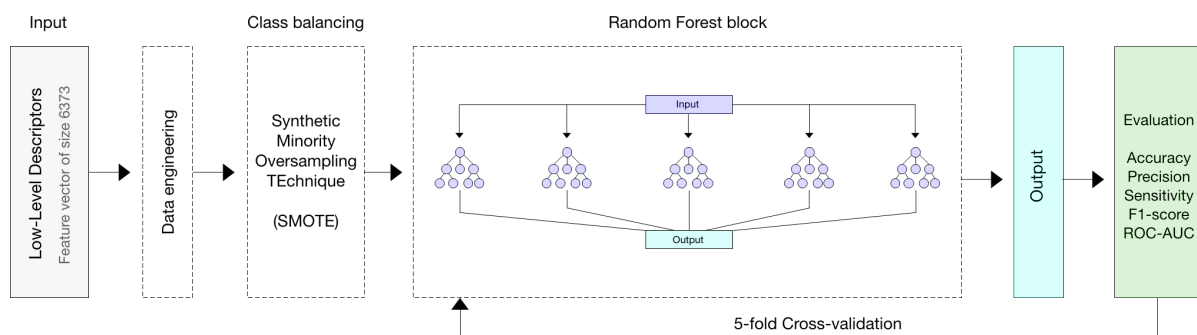


Figure 5.5: Systematic overview of the LLD-RF approach

<sup>2</sup><https://scikit-learn.org>



## 5.4.2 Results and discussion

We present the achieved results of the 5-fold Cross-Validation which took about 34 minutes of training, in terms of the aforementioned performance metrics, which uses the full-length LLD feature vector. Table 5.4 summarizes the test results of our classifier on the 20% of data. ROC curve, computed with predicted probabilities in Figure 5.7 (a) to summarize the test's overall diagnostic accuracy. Our model achieved a test ROC-AUC value of 0.9046 (0.952 using output probabilities), indicating that it has more than a 90% chance of discriminating between Likely-COVID-19 and Non-Likely-COVID-19 cases. The testing performance is illustrated as confusion matrix in Figure 5.7 (b), where only 7 samples belonging to Non-Likely-COVID-19 are misclassified. The model achieved more than 99% specificity for Non-Likely-COVID-19 cases, while its sensitivity was only 81.21% for Likely-COVID-19 cases.

Table 5.4: 5-fold Cross-Validation and testing results of the first approach LLD-RF

	Accuracy	Precision	Sensitivity	F1-Score	ROC-AUC	Specificity
Fold-1	89.77	99.31	80.24	88.76	89.84	99.44
Fold-2	88.61	98.36	78.68	87.43	88.68	98.67
Fold-3	89.92	99.44	80.44	88.93	89.99	99.54
Fold-4	89.09	99.18	78.97	87.93	89.15	99.33
Fold-5	89.34	98.87	79.72	88.27	89.40	99.08
Avg $\pm$ sd	89.35 $\pm$ 0.0047	99.03 $\pm$ 0.0038	79.61 $\pm$ 0.0068	88.26 $\pm$ 0.0054	89.41 $\pm$ 0.0047	99.21 $\pm$ 0.003
Test	<b>90.42</b>	<b>99.65</b>	<b>81.21</b>	<b>89.49</b>	<b>90.46</b>	<b>99.71</b>

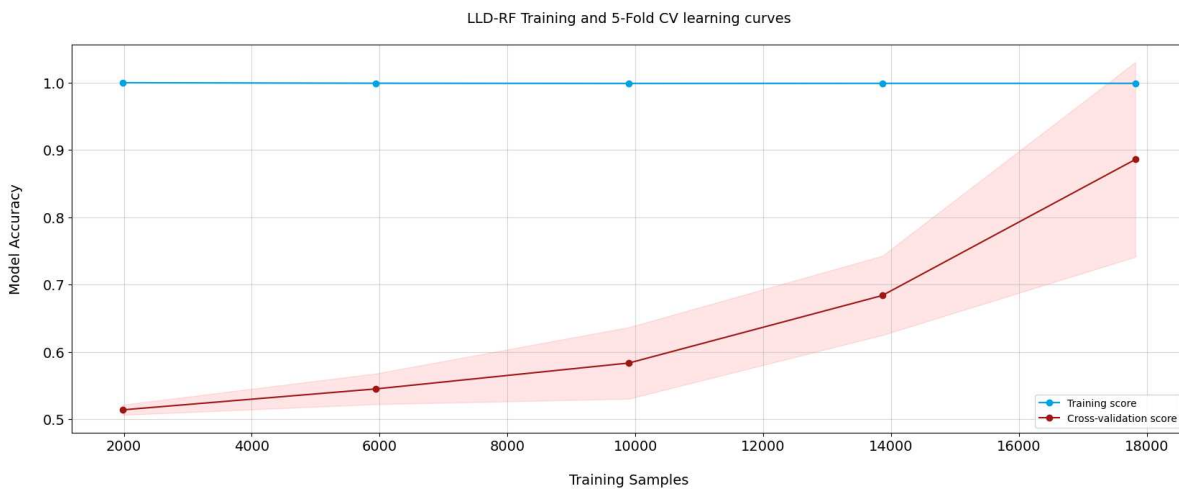


Figure 5.6: Training and Cross-Validation learning curves of first approach LLD-RF

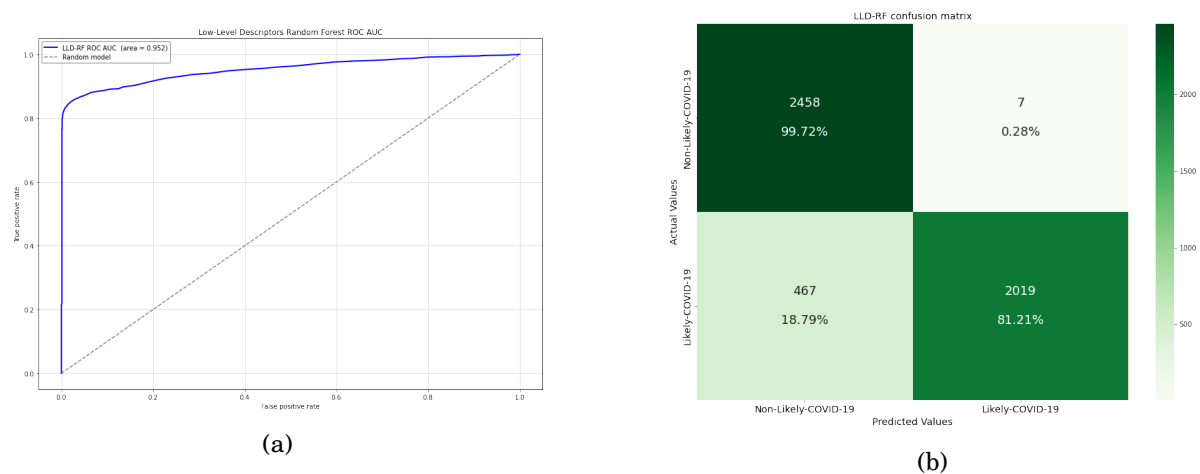


Figure 5.7: LLD-RF performance evaluation, (a) shows the ROC curve of the predicted probabilities and (b) presents the confusion matrix of the test set

The presented results of the test set for the Full-length LLD feature vector approach show an overall accuracy of 90.42% and 90.46% of ROC-AUC. The model demonstrates high specificity of 99.71%, which means that it has the ability to recognize almost all Non-Likely-COVID-19 cases. However, our focus is on the model’s sensitivity, where we denoted only 81.21% which is relatively lower than specificity, indicating that the model misclassifies about 19% of the Likely-COVID-19 cases in the test set. At this stage, we can conclude that our first approach suffers from a weakness related to sensitivity and can show critical cases (false negatives). Thus, further investigation is needed to improve the model’s performance in terms of sensitivity. Additionally, it is also important to note that the first approach uses all 6373 features in the feature vector, which can lead to possible overfitting and decrease the model’s generalization as illustrated in Figure 5.6, where we observe an increasing standard deviation after the initial training set of 2000 samples and this indicates that the model’s performance becomes less consistent. Therefore, feature selection or dimensionality reduction techniques can be applied to improve the model’s performance. Finally, it is worth noting that our model inference time is only 0.5 seconds for a test set of size 4951 samples, demonstrating the feasibility of our approach for real-time applications. Additionally, the model training time was very low, with only 7 minutes per fold, indicating the time efficiency of the approach.

## 5.5 Bottleneck-based feature extraction

In this section, we introduce a new approach that utilizes an autoencoder-based feature extraction technique to extract the most important features from the full-length LLD feature vector. The approach involves reducing the dimensionality from 6373 to 1024 and forwarding the resulting latent space to a Random Forest model. This approach is called Bottleneck-LLD-RF, and it aims

to improve the classification performance, specifically, the sensitivity of the model by selecting the most informative combination of features.

### 5.5.1 Methodology description and model training

Autoencoder is a neural network that compresses input data into a compact summary known as the *latent-space* representation  $z$ . The encoder, code, and decoder are the three parts of an autoencoder. The encoder compresses the input and generates the code, which the decoder subsequently uses to reconstruct the input using  $x' = wz + b$ . As proposed in our study, autoencoders can be used as a dimensionality reduction technique, which reduces and compresses the initial feature vector while preserving the meaningful patterns inside. In order to train our autoencoder network, we used two stacked encoder/decoder layers. The first encoder layer, activated by ReLU function, has 2048 units and receives our extracted LLD feature vector of size 6373. The result is then forwarded to the second encoder layer with 1024 units, which represents our bottleneck or latent space's size and the final size of our new feature vector. For the decoder block, the first decoder layer consists of 2048 units and receives the bottleneck layer, followed by the last decoder layer to map the hidden representation  $z$  with the original data  $x$  of size 6373. Our autoencoder network was trained for 100 epochs with a batch size of 256 using Adam optimizer and BCE the loss function. Figure 5.8 shows a general overview of our approach architecture. For Random Forest classifier stage and also the autoencoder model, we tuned several hyper-parameters through GridSearch to find the best configuration for our approach, Table 5.5 summarizes the tuned parameters where *split\_quality* presents the function of measurement of data split quality.

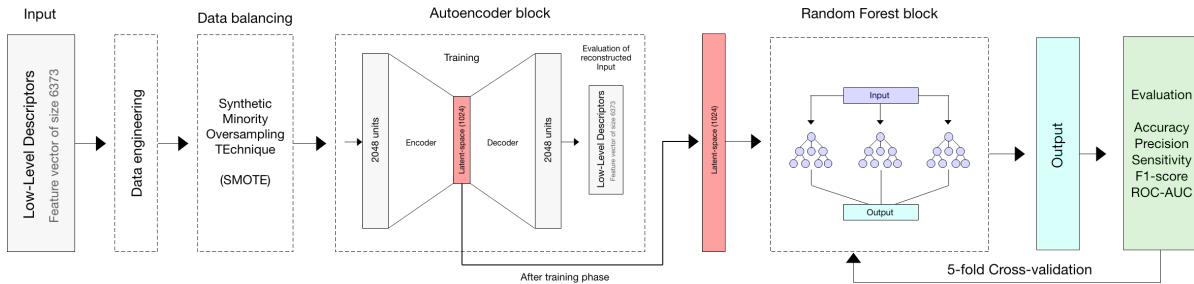


Figure 5.8: Block diagram of the second approach Bottleneck-LLD-RF

### 5.5.2 Results and discussion

Figure 5.9 presents reconstruction training and validation losses for our autoencoder model. The train/validation loss curves indicate how well our autoencoder network is performing in terms of reconstruction loss over the training and validation sets. In this case, the smoothness and absence of perturbations in the curves suggest that the model is not overfitting or underfitting the data, and it is able to generalize well to unseen data. The absence of a significant gap between the train and validation curves suggests that our autoencoder model is not overfitting.

Table 5.5: Tuned hyper-parameters for the second approach Bottleneck-LLD-RF where  $\sqrt{F'}$  presents the square root of the latent space (1024)

	Tested values	Best value
n_estimators	[200-1200]	<b>1100</b>
bootstrapping	{Used, Not used}	<b>Used</b>
max_features	$\{F', \sqrt{F'}\}$	$\sqrt{F'}$
split_quality	{Gini, Entropy}	<b>Gini</b>

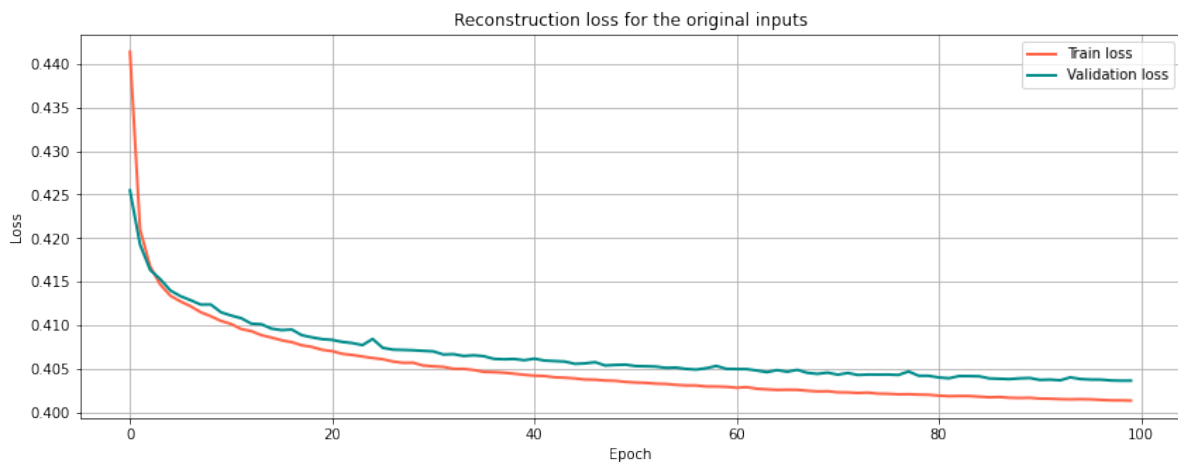


Figure 5.9: Autoencoder reconstruction loss curves for training and validation

Table 5.6 presents the Cross-Validation and test results of our classifier on the 20% of data for the second approach, bottleneck-based feature extraction using autoencoder. The ROC curve, computed with predicted probabilities, is presented in Figure 5.11 (a) to summarize the test's overall diagnostic accuracy, where Figure 5.11 (b) shows the confusion matrix for the testing set. Compared to the first approach, we denote a small improvement of the overall accuracy and ROC-AUC score with 0.22% and 0.20%, respectively, where a significant improvement of the model's sensitivity in the test set is denoted by 5.07%. On the other hand, our model yielded a degradation of 5.03% and 4.67% for precision and specificity, respectively.

Table 5.6: 5-fold Cross-Validation and testing results of the second approach Bottleneck-LLD-RF

	Accuracy	Precision	Sensitivity	F1-Score	ROC-AUC	Specificity
Fold-1	87.09	92.57	80.63	87.09	89.84	95.35
Fold-2	88.41	93.62	82.40	88.40	88.68	94.40
Fold-3	86.94	93.87	79.01	86.93	89.99	94.85
Fold-4	88.43	93.47	82.60	88.42	89.15	94.24
Fold-5	87.37	92.66	81.14	87.36	89.40	93.59
Avg $\pm$ sd	87.65 $\pm$ 0.0064	93.24 $\pm$ 0.0052	81.16 $\pm$ 0.0130	86.77 $\pm$ 0.0077	87.64 $\pm$ 0.0064	94.12 $\pm$ 0.0049
Test	<b>90.64</b>	<b>94.62</b>	<b>86.28</b>	<b>90.26</b>	<b>90.66</b>	<b>95.04</b>

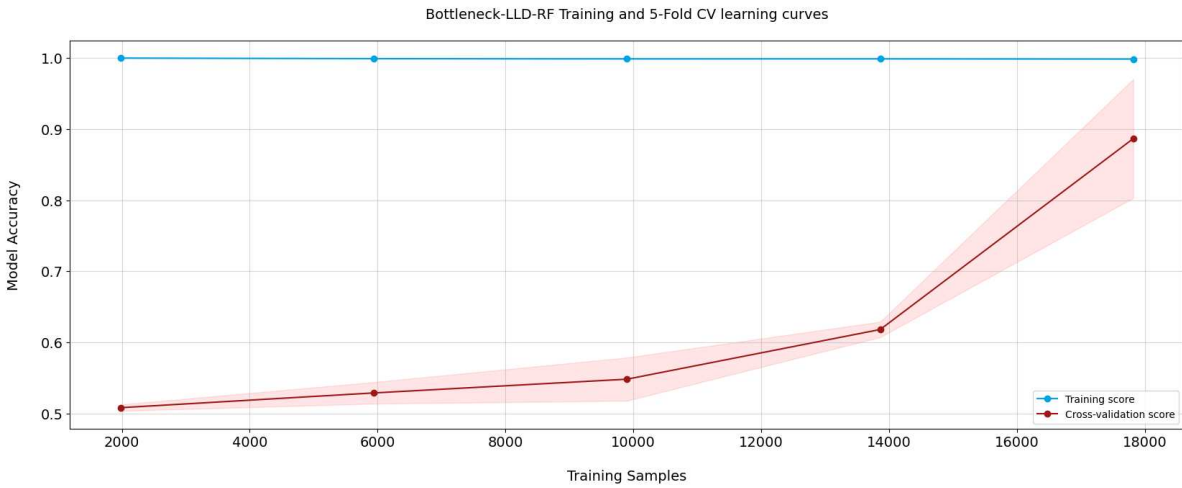


Figure 5.10: Training and Cross-Validation learning curves of second approach Bottleneck-LLD-RF

The second approach improved ROC-AUC for the predicted classes by only 0.2%, indicating a slight improvement in the discrimination between Likely-COVID-19 and Non-Likely COVID-19 samples. However, when computing the ROC curve using predicted probabilities, the proposed system obtains almost the same AUC value as the previous approach (0.951) as well as the ROC curve, if we observe ROC curves in Figure 5.7 (a) and Figure 5.11 (b), we can note that they are almost identical due to the improvement of the system’s sensitivity and degradation of the specificity and precision. Regarding learning process, similar to LLD-RF, the validation curve of Bottleneck-LLD-RF also shows an increasing performance as the number of samples in the training set increases, as shown in Figure 5.10. However, compared to LLD-RF, it exhibits a smaller standard deviation. This can be attributed to the use of bottleneck features, by reducing the dimensionality of the feature space, the model had a more stable and consistent performance, leading to a smaller standard deviation. We shall denote that our autoencoder training was performed once for 42 minutes, while training the Random forest classifier for 6.5 minutes per

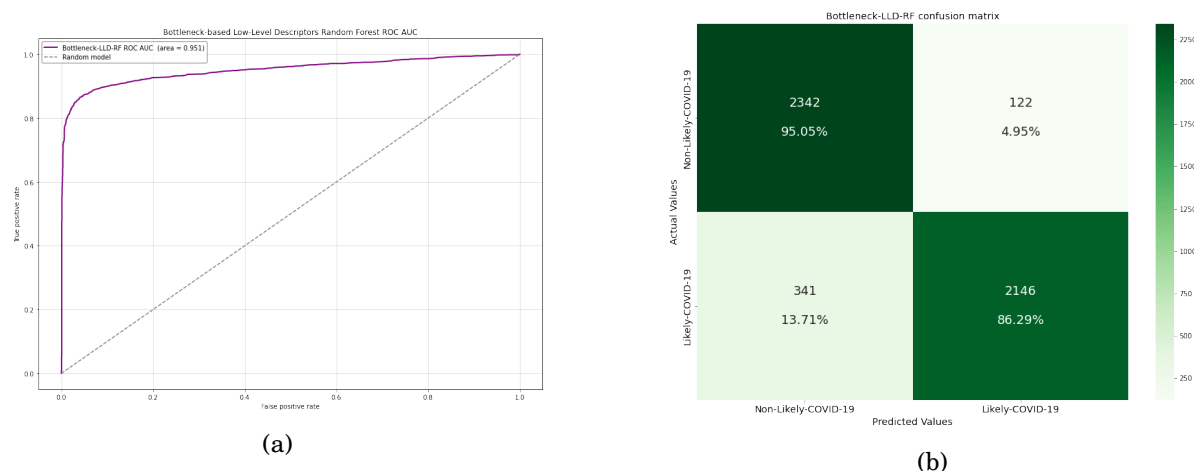


Figure 5.11: Bottleneck-LLD-RF performance evaluation, (a) shows the ROC curve of the predicted probabilities, and (b) presents the confusion matrix of the test set

fold, with an inference time of 1.8 seconds for a test set of the same size.

## 5.6 LLE dimensionality reduction

Our hypothesis (Section 5.2.3) is that dimensionality reduction can lead to a more discriminative combination of features and thus enhance the classification performance. To validate this, we propose a third approach, named LLE-LLD-RF, which utilizes LLE as a dimensionality reduction technique to extract the most informative features from the full-length LLD feature vector. The reduced feature set is then fed to the Random Forest model for classification.

### 5.6.1 Methodology description and model training

LLE is a dimensionality reduction technique that can reveal the intrinsic low-dimensional structure of high-dimensional data by preserving the local relationships between data points. The main idea behind LLE is to represent each data point as a weighted linear combination of its  $k$  nearest neighbors in the original high-dimensional space. Then, the low-dimensional representation of each data point is found by minimizing the reconstruction error  $e(Y) = \sum_i |y_i - \sum_{j=1}^n w_{ij}y_j|^2$ , where  $W$  is the pre-optimized matrix of weights that maps data points with their  $k$  nearest neighbors. The aim is to use the same weight matrix to find the projection  $Y$  where  $y_i$  presents the data point  $x_i$  in the low-dimensional space (see Section 2.8.1). The main advantage of LLE over other linear and non-linear techniques is that it is able to capture and preserve the local structure and non-linear relationships within high-dimensional data. In our approach, we applied LLE to our full-length LLD feature vector, reducing its dimensionality from 6373 to 3187, which is the half of the original dimensionality. The number of components was selected based on a trade-off between computational complexity and preserving the most informative patterns in

the data. The reduced feature vectors were then forwarded to the Random Forest classifier for training and evaluation. The same strategy of choosing the best hyper-parameters was followed, particularly, the number of neighbors  $k$  of the original high-dimensional space was set to 5, as shown in Table 5.7. The architecture of the third approach is depicted in Figure 5.12 providing a general overview.

Table 5.7: Selected hyper-parameters for the third approach LLE-LLD-RF where  $\sqrt{F'}$  presents the square root of the LLE reduced vector (3187)

	Tested values	Best value
n_estimators	[200-1200]	<b>1100</b>
bootstrapping	{Used, Not used}	<b>Used</b>
max_features	{ $F'$ , $\sqrt{F'}$ }	$\sqrt{F'}$
split_quality	{Gini, Entropy}	<b>Gini</b>
$k$	[2-10]	<b>5</b>

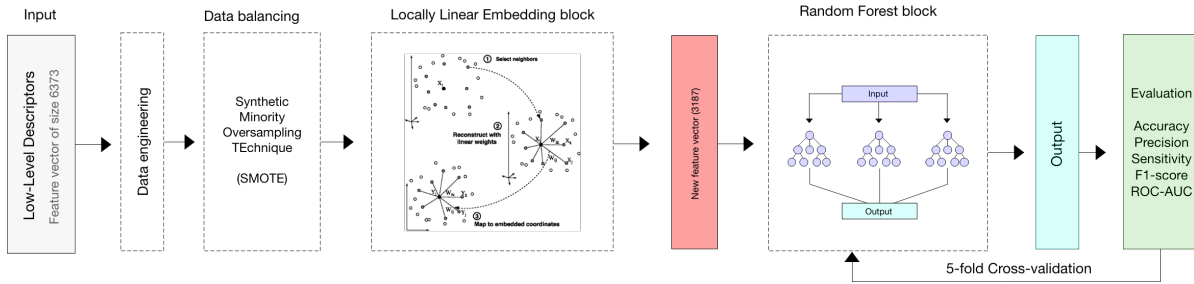


Figure 5.12: General overview of the LLE-LLD-RF

## 5.6.2 Results and discussion

Table 5.8 shows the 5-fold Cross-Validation and test results of our third approach LLE-LLD-RF, where we observe a significant testing improvement of the overall accuracy, sensitivity, and ROC-AUC by 2.87%, 7.04%, and 2.85%, respectively, compared to the previous approach. The improvement of ROC-AUC by 2.85% (0.975 for predicted probabilities as shown in Figure 5.14 (a)) indicates that the model is able to distinguish between Likely-COVID-19 and Non-COVID-19 with higher performance than LLD-RF and Bottleneck-LLD-RF. This approach shows a very significant enhancement of the sensitivity against Bottleneck-LLD-RF, from 86.28% to 93.89%, which means that our new model LLE-LLD-RF was able to recognize and correctly classify almost 94% of Likely-COVID-19 cases in the testing set (2333 against only 168 which were misclassified as shown in Figure 5.14 (b)). In contrast, we observe that our third approach LLE-LLD-RF shows a slight decrease of 1.82% and 1.3% in terms of specificity and precision, respectively. The validation curve of LLE-LLD-RF shows a similar increasing performance trend as the previous

approaches. However, the interesting observation here is the behavior of the standard deviation. Unlike the other approaches, the standard deviation starts with high values and gradually decreases as the training sample size increases (see Figure 5.13). This suggests that initially, the model’s performance varies significantly between different subsets of the training data. As more data points are included in the training, the model learns more representative patterns and becomes more consistent in its predictions, resulting in a smaller standard deviation. The training time of LLE-LLD-RF approach was about 60 minutes (including 37 minutes for fitting LLE), with an inference time of 2.3 seconds for the same testing set.

Table 5.8: 5-fold Cross-Validation and testing results of the third approach LLE-LLD-RF

	Accuracy	Precision	Sensitivity	F1-Score	ROC-AUC	Specificity
Fold-1	92.72	92.15	93.37	92.76	92.72	92.08
Fold-2	92.47	91.91	93.12	92.51	92.47	91.83
Fold-3	91.74	90.84	92.82	91.82	91.74	90.67
Fold-4	92.34	92.31	92.36	92.34	92.34	92.33
Fold-5	91.94	92.29	91.50	91.90	91.94	92.38
Avg $\pm$ sd	92.24 $\pm$ 0.0035	91.90 $\pm$ 0.0055	92.63 $\pm$ 0.0065	92.27 $\pm$ 0.0036	92.24 $\pm$ 0.0035	91.85 $\pm$ 0.0062
Test	<b>93.51</b>	<b>93.32</b>	<b>93.89</b>	<b>93.56</b>	<b>93.51</b>	<b>93.22</b>

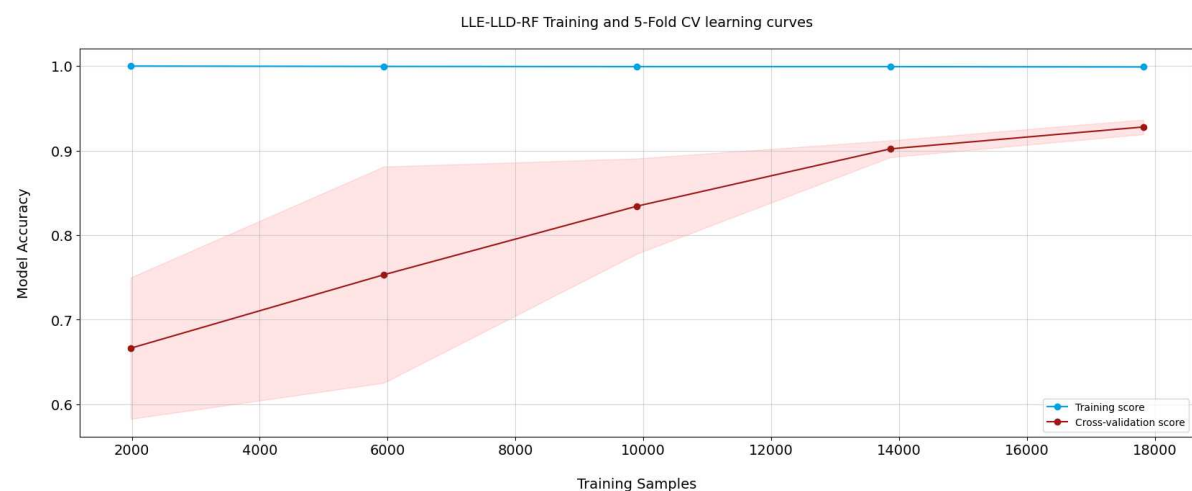


Figure 5.13: Training and Cross-Validation learning curves of third approach LLE-LLD-RF



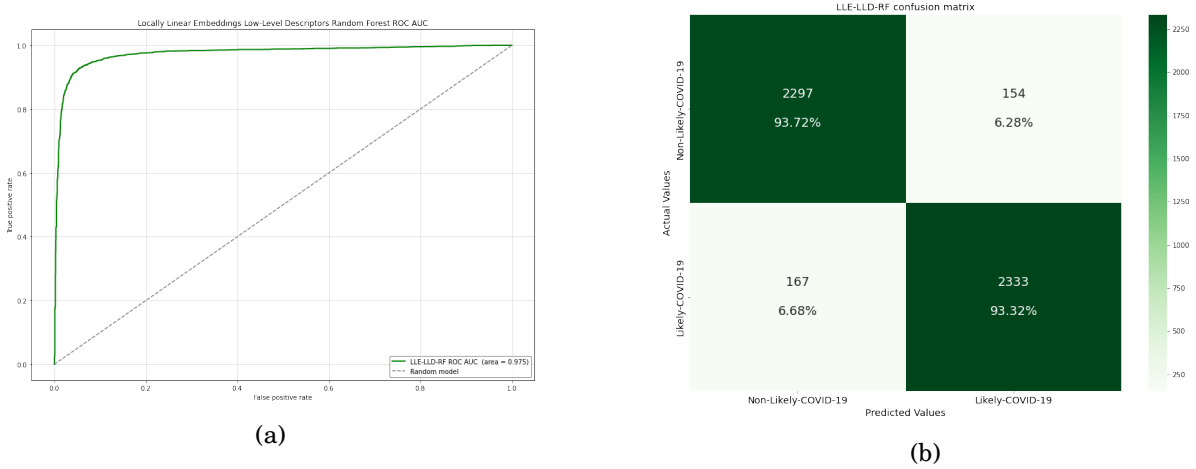


Figure 5.14: The performance evaluation of the LLE-LLD-RF approach includes a ROC curve for the predicted probabilities and a confusion matrix for the test set. The ROC curve is shown in (a), while the confusion matrix is presented in (b)

## 5.7 Analysis and comparison

Based on the results presented in Tables 5.4, 5.6, and 5.8, we can observe that all approaches achieved an acceptable overall accuracy. However, there are some slight variations in their performance, leading us to favor one approach over the others. We present a summary and comparison of the achieved results in Table 5.9.

Table 5.9: Comparison of 5-fold CV and testing results of the three approaches, the first block of the table presents the Cross-Validation results, shown as  $avg \pm sd$ , while the second block shows testing results on the same testing set. The best testing results are highlighted with bold text

Approaches	Evaluation Metrics					
	Accuracy	Precision	Sensitivity	F1-Score	ROC-AUC	Specificity
LLD-RF	89.35 $\pm$ 0.004	99.03 $\pm$ 0.003	79.61 $\pm$ 0.006	88.26 $\pm$ 0.005	89.41 $\pm$ 0.004	99.21 $\pm$ 0.003
Bottleneck-LLD-RF	87.65 $\pm$ 0.006	93.24 $\pm$ 0.005	81.16 $\pm$ 0.0130	86.77 $\pm$ 0.007	87.64 $\pm$ 0.006	94.12 $\pm$ 0.005
LLE-LLD-RF	92.24 $\pm$ 0.0035	91.90 $\pm$ 0.0055	92.63 $\pm$ 0.0065	92.27 $\pm$ 0.0036	92.24 $\pm$ 0.003	91.85 $\pm$ 0.006
LLD-RF [44]	90.42	<b>99.65</b>	81.21	89.49	90.46	<b>99.71</b>
Bottleneck-LLD-RF [137]	90.64	94.62	86.28	90.26	90.66	95.04
LLE-LLD-RF [136]	<b>93.51</b>	93.32	<b>93.89</b>	<b>93.56</b>	<b>93.51</b>	93.22

Regarding Cross-Validation results, it is important to note that standard deviation values were in general, below 0.01, indicating that our models had relatively consistent performance across the different folds of the Cross-Validation. However, we observe the highest value  $sd = 0.0130$  for the sensitivity of Bottleneck-LLD-RF which suggests that the model had more variability

in Likely-COVID-19 cases identification. On the other hand, typically, Cross-Validation results are close to the testing results, indicating that the models are not overfitting and can generalize to unseen data. Regarding testing performance, first, LLD-RF approach achieved the lowest sensitivity among the three approaches in the testing phase, with a sensitivity of 81.21% which leads us to conclude that LLD-RF had not the ability to perform well with Likely-COVID-19 class. However, it showed the highest precision (99.65%) and specificity (99.71%), which suggests that LLD-RF approach is more conservative and not well sensitive to predict Likely-COVID-19 cases, where 18.79% of Likely-COVID-19 test samples (467/2486) are misclassified. Second, Bottleneck-LLD-RF showed a lower specificity which means that this model is not able to correctly classify more samples from Non-Likely-COVID-19 compared to LLD-RF. In contrast, Bottleneck-LLD-RF exhibited a higher sensitivity, compared to LLD-RF with a 5.07% increase due to the use of the two-layer autoencoder, which had the ability to compact the LLD feature vector and extract more patterns that are related to Likely-COVID-19 class, the achieved performance of 86.28% may not be considered reliable in medical diagnosis cases, where 13.71% of Likely-COVID-19 test cases (341/2486) are not correctly recognized and classified. For further investigation and in order to validate our aforementioned hypothesis (Section 5.2.3), a third approach based on dimensionality reduction has been developed and validated. LLE-LLD-RF approach achieved the highest sensitivity among the three approaches, with a considerable improvement compared to the other two approaches, 93.89% compared to 81.21% and 86.28%. Also, the overall accuracy has reached 93.51% outperforming 90.42% and 90.64% for LLD-RF and Bottleneck-LLD-RF, respectively. However, regarding specificity and precision, this approach showed the lowest performance, 93.22%, 93.32% compared to 99.71%, 95.04%, and 99.65, 94.62%. Thus, the choice of the best approach is based on a compromise between sensitivity and specificity with a particular focus on sensitivity due to its importance in the present context. Figure 5.15 presents a single plot that shows the ROC curves of the three approaches, ROC curves provide a graphical representation of the performance of each approach in terms of the trade-off between sensitivity and specificity. The closer the curve is to the top-left corner, the better the overall performance. From the plotted ROC curves, it is clear that LLE-LLD-RF outperforms the other two approaches, as its curve is higher and closer to the top-left corner. The curves of LLD-RF and Bottleneck-LLD-RF appear to be quite similar, indicating that their overall performance is also similar. However, the ROC-AUC metric provides a quantitative measure of the overall performance of the approaches, taking into account the entire range of the ROC curve. The results show that LLE-LLD-RF achieved the highest ROC-AUC value (0.975 for predicted probabilities), followed by LLD-RF and Bottleneck-LLD-RF (0.952 and 0.951). Therefore, the ROC curves and ROC-AUC metric confirm the superiority of LLE-LLD-RF over the other approaches in terms of overall performance.

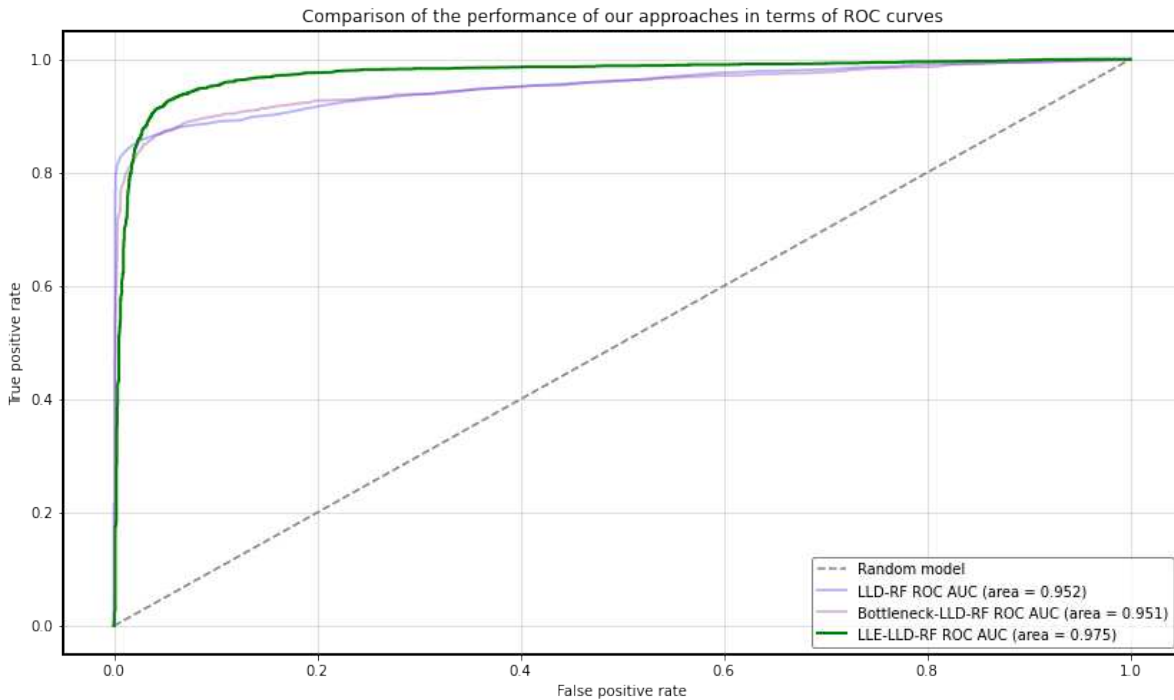


Figure 5.15: ROC curves comparison for the three approaches

The obtained testing results confirm our hypothesis that dimensionality reduction may lead to combine and extract the most informative patterns, hence improving the performance of the classification model. LLE-LLD-RF approach achieved the highest performance with an accuracy of 93.51%, a sensitivity of 93.89% and ROC-AUC of 93.51% (0.975 for predicted probabilities), outperforming both LLD-RF and Bottleneck-LLD-RF approaches. This improvement is attributed to the fact that LLE preserves the local structure of the data while reducing its dimensionality, while the latent space in Bottleneck-LLD-RF approach may discard some useful information in the process of compacting and reducing dimensionality in the neural network. In terms of time efficiency, Table 5.10 summarize the time cost performance for training and inference of the three approaches. The training time is measured in seconds and represents the time it takes to train the model on the dataset (time to perform 5-fold Cross-Validation), while the inference time is also measured in millisecond and represents the time it takes to classify one single instance using the trained model.

Discussing Table 5.10, LLD-RF has the shortest training and inference time among the three approaches, with only 2040 seconds for training and 0.1 milliseconds for inference due to the small number of Random Forest estimators. Bottleneck-LLD-RF has a longer training and inference time compared to LLD-RF, with 4470 seconds for training and 0.36 millisecond for inference, and this can be explained by the complexity of computing in the encoder and decoder networks (2500 seconds in the training phase). LLE-LLD-RF has the shortest training time among the three approaches, with 1380 seconds, however, the remaining time (2243 seconds) was spent to

Table 5.10: The time cost of the three approaches for training (explained in seconds) and inference (explained in millisecond)

<b>Approach</b>	<b>Training time</b>	<b>Inference time</b>
LLD-RF	2040	0.1
Bottleneck-LLD-RF	4470	0.36
LLE-LLD-RF	3623	0.46

fit LLE with original data which resulted in a total of 3623 seconds, and also, has the longest inference time of 0.46 millisecond, due to the computation of the new embedded feature vector. Considering the performance results, LLE-LLD-RF outperformed both LLD-RF and Bottleneck-LLD-RF approaches regarding accuracy, sensitivity, and ROC-AUC. However, LLE-LLD-RF has a longer inference time compared to the other two approaches. On the other hand, LLD-RF has the shortest time for both training and inference, but it has lower accuracy and sensitivity compared to LLE-LLD-RF. Therefore, considering the medical context of the problem, where sensitivity is of utmost importance, the choice of the best approach is mainly related to the factor of the system's performance, thus, LLE-LLD-RF would be the best approach to be implemented. This conclusion is supported by the higher sensitivity achieved by LLE-LLD-RF compared to the other two approaches, as well as its overall better performance and comparable inference time. Although LLE-LLD-RF has a shorter training time than Bottleneck-LLD-RF, it has a slightly longer inference time, which may be negligible in practical use cases. Therefore, LLE-LLD-RF is the recommended approach for medical diagnosis based on the discussed performance results and time information.

## 5.8 Ablation study

In this section, we present an ablation study that aims to demonstrate the effectiveness of the proposed approach (we consider the best approach LLE-LLD-RF), where we conducted extensive additional experiments to illustrate the impact of the used classifier (Random Forest) and dimensionality reduction (LLE). This study is divided into three parts, each focusing on a specific aspect of the ablation. First, we explore the effectiveness of Random Forest and ensemble classifiers against several ML algorithms, highlighting their strengths and weaknesses in our context. Second, we investigate the effectiveness of LLE, by replacing it with other dimensionality reduction techniques, and third, we conduct a feature analysis to demonstrate the crucial influence of dimensionality reduction on the overall classification performance.

### 5.8.1 Effectiveness of Random Forest and Ensemble classifiers

We evaluated the performance of various simple and ensemble classifiers, in order to assess the effectiveness of Random Forest and ensemble-based approaches. The classifiers were trained and evaluated on the same testing set to ensure consistent results and comparison. Table 5.11 highlights the performance achieved by 14 different classifiers, from different categories including ensemble-based, Discriminant Analysis (DA)-based, and support vector-based.

Table 5.11: First part of the ablation study which involves training different classifiers to demonstrate the performance of the presented study.

Classifier	Evaluation Metrics					
	Accuracy	Precision	Sensitivity	F1-Score	ROC-AUC	Specificity
Logistic Regression	64.71	70.29	51.48	59.43	64.77	78.05
XGBoost	93.19	94.09	92.23	93.15	93.19	94.15
kNN	70.69	93.08	44.97	60.64	70.80	96.63
Linear SVM	64.75	79.47	40.18	53.37	64.85	89.53
Kernel SVM	91.41	95.13	87.36	91.08	91.43	95.49
AdaBoost	63.21	71.74	44.12	54.64	63.30	82.47
Naive Bayes	89.29	95.44	82.62	88.57	89.32	96.02
MLP	78.73	80.32	76.34	78.28	78.74	81.13
Decision Trees	80.12	79.56	81.29	80.42	80.12	78.94
GB	92.24	92.00	92.59	92.30	92.24	91.88
Extra-Trees	93.17	94.97	91.23	93.06	93.18	95.13
Quadratic DA	79.60	87.12	69.67	77.42	79.64	89.61
LDA	65.34	69.50	55.18	61.52	65.38	75.57
RBFSampler	87.63	93.82	80.69	86.76	87.66	94.64
Random Forest	93.51	93.32	93.89	93.56	93.51	93.22

Table 5.11 presents the results of training various classifiers and evaluating their performance. In terms of overall accuracy, it can be observed that only Kernel SVM and ensemble-based methods (XGBoost, Extra-Trees, Random Forest, and GB) achieved the highest overall performance. Examining the precision metric, several classifiers achieved more than 92% precision. However, most of these classifiers obtained relatively lower accuracy and sensitivity, for instance, kNN with 93.08% of precision, 70.69% and 44.97% of sensitivity, and Naive Bayes, which recorded the highest precision of 95.44% and only 82.62%. This phenomenon suggests that although these classifiers can accurately predict positive instances, they struggled to identify all the positive instances in the dataset, leading to low sensitivity and overall accuracy. Moving on to sensitivity, only four classifiers (ensemble-based methods: GB, Extra-Trees, Random Forest, and XGBoost) achieved more than 90% whereas other methods ranged from 44.12% to 87.36%. This indicates their effectiveness in discovering and capturing complex patterns related to Likely-COVID-19 class. The same methods, along with Kernel SVM, demonstrate high performance in the aspect of

distinguishing between Likely-COVID-19 and Non-Likely-COVID-19, as indicated by their high ROC-AUC values. Regarding specificity, we observe varying results across different classifiers. Kernel SVM, Naive Bayes, kNN, and most ensemble-based methods, achieved more than 93% indicating the high capacity of recognizing Non-Likely-COVID-19 samples, however, by giving a particular focus on the sensitivity, the choice of the best method is questioned, for instance, considering Naive Bayes, its ability to recognize Non-Likely-COVID-19 class was achieved 96.02% where only 82.62% of sensitivity was obtained. This study showed the need to focus on the overall model’s performance, in other words, finding a trade-off between sensitivity, precision, and specificity. Thus, by considering the above notes, the best performing methods are the ensemble-based ones, in particular, Random Forest with an overall accuracy of 93.51% and 93.89% of sensitivity.

### 5.8.2 Effectiveness of LLE

In this section, we investigate the effectiveness of LLE, by comparing it with several other methods. Specifically, we explore the performance of LLE, in comparison to PCA, LDA, Isomap, t-SNE, Singular Value Decomposition (SVD), and Random Projection. By using these alternative methods, we aim to showcase the unique strengths of LLE. Throughout all experiments, we maintain the same configuration of Random Forest classifier to ensure a fair and consistent comparison.

Table 5.12: Second part of the ablation study which involves training our Random Forest classifier using the embeddings and components generated by 6 dimensionality reduction techniques.

Method	Evaluation Metrics					
	Accuracy	Precision	Sensitivity	F1-Score	ROC-AUC	Specificity
PCA	90.40	97.70	82.90	89.70	90.50	98.00
LDA	80.80	81.00	80.60	80.80	80.80	80.90
Isomap	69.37	67.97	73.77	70.75	69.36	64.94
t-SNE	81.31	77.12	89.25	82.75	81.28	73.30
SVD	90.50	98.00	82.80	89.80	90.60	98.30
Random Projection	90.97	97.93	83.78	90.30	91.00	98.21
LLE	93.51	93.32	93.89	93.56	93.51	93.22

Table 5.12 presents the results obtained from applying various dimensionality reduction techniques and training the Random Forest classifier using the resulting embeddings or components. When comparing LLE to other dimensionality reduction techniques, several observations can be made. While PCA, SVD, and Random Projection achieve accuracy levels near that of LLE (90.40%, 90.50%, and 90.97% respectively), their sensitivity falls behind. This indicates that these methods might not capture the intricate relationships within the data that are important for identifying

positive instances accurately. Although they may achieve high precision and specificity, their limited ability to identify true positives affects their overall performance. LDA, on the other hand, achieved a balanced performance and shows comparable sensitivity to PCA, SVD, and Random Projection. However, its overall performance is noticeably lower (80.80%). LDA operates based on linear projections that were not able to fully capture the complex nonlinear patterns of the data, leading to the mentioned suboptimal performance. While Isomap recorded the lowest performance among all methods, t-SNE, achieves a higher sensitivity (89.25%), however, a lower overall performance of only 81.31% has been recorded. Considering our criteria of prioritizing the sensitivity metric, LLE emerges as the top performer among the evaluated dimensionality reduction techniques. It successfully identifies a high percentage of true positives, making it particularly suitable for applications where correctly detecting positive instances is critical. Furthermore, LLE achieves a balanced performance by maintaining competitive precision, indicating its ability to distinguish between Likely-COVID-19 and Non-Likely-COVID-19 instances with the higher possible performance.

### 5.8.3 Reduced feature analysis

We delve into the analysis of reduced features obtained through autoencoder and LLE. We begin by visualizing the 3D scatter plots of the three best features among each of the original, latent space, and LLE features, ranked by Random Forest importance. Figure 5.16 showcases the 3D scatter plots, where each data point is represented by its three best features. Despite selecting the best discriminating features, the visual interpretation of these plots is challenging due to the complexity and overlapping nature of the data points. Additionally, Figures 5.16 (b) and Figure 5.16 (c) show a complex data structure although reduction with autoencoder and LLE. The 3D scatterplots exhibit a high concentration of data points from both classes in one region, making it difficult to discern clear patterns or class separation. This highlights the limitations of visual analysis in complex data even after applying dimensionality reduction, emphasizing the need for alternative visualization techniques.

To address the limitation of the scatterplots, we propose an alternative method, density plots, that make, in addition to the aforementioned performance, insights about the role of dimensionality reduction in the process of finding a discriminating combination of features that helps build efficient classification models. Specifically, we used KDE, which facilitates a more accurate depiction of the data by providing a smooth and continuous estimation of the distribution. These density plots highlight the distribution of the top five features ranked by Random Forest importance, and illustrated in Figure 5.17.

By examining the density distribution of the original features, we observed that it is not clear in distinguishing between the two classes. The density values showed almost the same level of density for both classes, a heavy overlap between the distributions, and mixed density peaks. This indicates that the original feature set alone may not be sufficient to separate classes effectively.

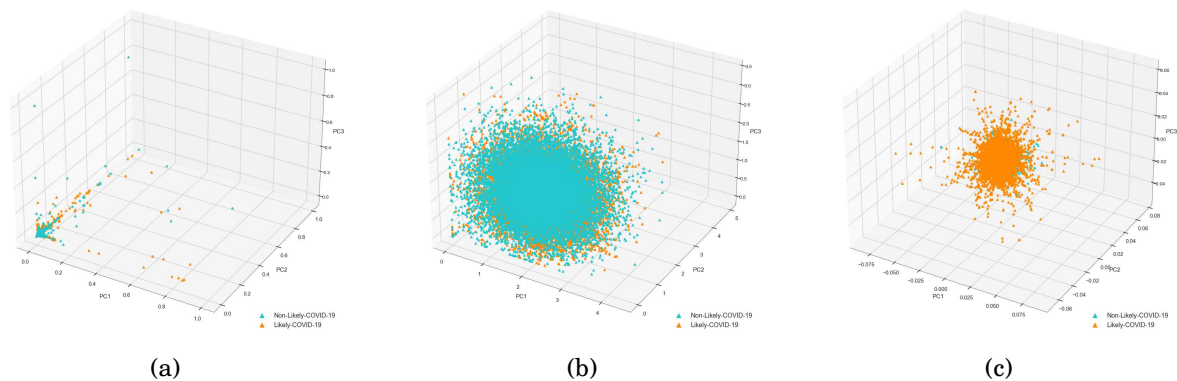


Figure 5.16: 3D scatter plots for the three best features, ranked by Random Forest importance, selected from original (a), latent space (b), and LLE features (c)

On the other hand, the application of the autoencoder for feature compression did not well improve the separability where we observe the presence of a high overlap between the two classes. Moreover, LLE feature set demonstrated its effectiveness in capturing features that yielded a significant trend in density values. We observed a notable trend of high density for Non-Likely-COVID-19 class and low density for Likely-COVID-19 class. While it may be challenging to directly interpret this as class separability based on the plotted features alone, these features can potentially contribute to further combinations that are valuable for distinguishing between the two classes. Furthermore, we noticed that there were specific regions in the density plots where many data points from the Non-Likely-COVID-19 class were concentrated. This suggests that the LLE features may be capturing specific patterns or subgroups within the class, which could be further explored to enhance the classification performance. While the density plots provide insights into the distribution of feature values and show some degree of separability between the classes, it is important to note that the complexity of the data may limit the extent of class separability. The observed trends in density values indicate potential discriminative patterns, but further investigation is necessary to fully assess the separability of the classes. In the previous sections, we evaluated the performance of LLE using various evaluation metrics, which demonstrated its effectiveness in improving classification accuracy, precision, sensitivity, F1-score, and specificity. These results reinforce the notion that ML models can leverage the extracted features to enhance class separability. Therefore, while the density plots provide valuable visual insights, their interpretation should be complemented by the results obtained from rigorous ML modeling and performance evaluation as aforementioned in previous sections.

## 5.9 Concluding notes

ML approaches based on ensemble learning for COVID-19 screening have been investigated in this chapter. Our research question focused on whether ML methods can achieve comparable



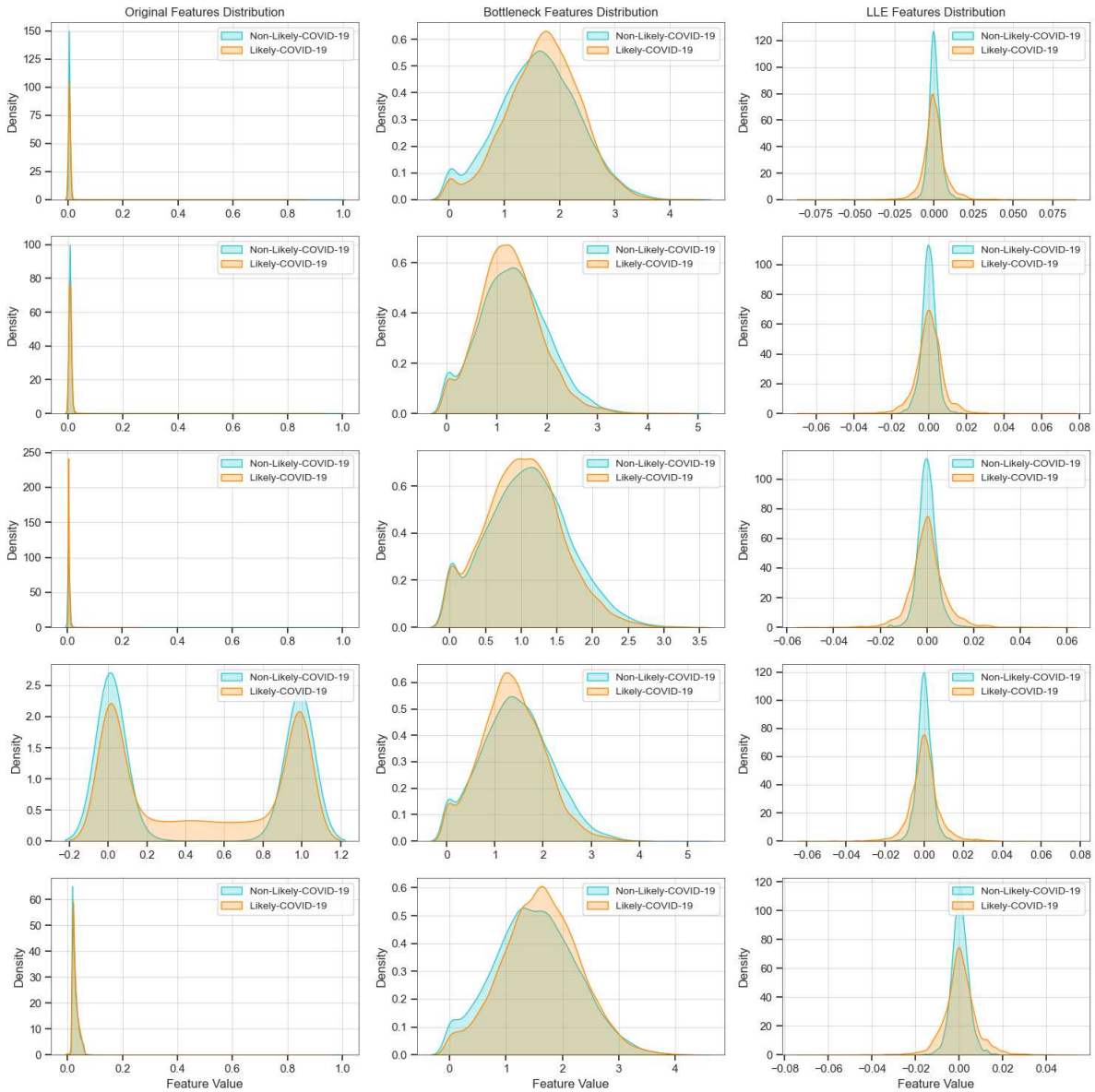


Figure 5.17: Density distribution plots of the top 5 features, ranked in descending order of Random Forest importance, for the original, latent space, and LLE features. The plots illustrate the distribution of feature values using KDE for the Non-Likely-COVID-19 and Likely-COVID-19 classes.

performance to DL methods in developing accurate and efficient diagnosis systems. Although DL approaches have been considered the most effective methods for analyzing complex data, we have shown and demonstrated that ML methods can also achieve high performance in the present context. In the literature review, we discussed the performance achieved using ML methods for COVID-19 diagnosis. Although the results varied widely, some studies reported high accuracy and sensitivity, indicating the potential of ML methods for COVID-19 diagnosis. However, the

performance of ML methods is strongly dependent on several factors, such as the quality and size of the dataset, the extracted and selected features, and also, classification methods. In our study, we focused on three ML approaches based on ensemble learning: LLD-RF, Bottleneck-LLD-RF, and LLE-LLD-RF. We evaluated the performance of these approaches using COUGHVID dataset, which was balanced using SMOTE method. We used 5-fold Cross-Validation and a separate testing set to evaluate the performance of the approaches. Our results have shown that LLE-LLD-RF approach achieved the highest performance with an accuracy of 93.51%, a sensitivity of 93.89%, and a ROC-AUC of 93.51% (97.5% for predicted probabilities), outperforming both LLD-RF and Bottleneck-LLD-RF approaches. This improvement is attributed to the fact that LLE preserves the local structure of the data based on the assigned neighbors' weights while searching for the optimal projection  $Y$ , while the latent space in the Bottleneck-LLD-RF approach discarded some important patterns in the process of compacting which inevitably causes information loss, and that was confirmed by our downstream task performance which has shown the superiority of LLE-LLD-RF, specifically, in terms of sensitivity. As already mentioned, the sensitivity is the most important metric in our context, as it measures the ability of the model to correctly identify positive cases and avoid critical cases (false negatives). LLE-LLD-RF approach has shown the highest sensitivity among the three approaches, making it the appropriate choice for this application. In addition, we evaluated the time complexity of the three approaches for training and inference. LLE-LLD-RF approach showed a slightly longer inference time (0.46 millisecond per instance) compared to LLD-RF and Bottleneck-LLD-RF, however, an inference time of 0.46 millisecond per instance is considered highly efficient for real-time use, which makes our solution a time-effective and can be deployed for further production use. Our findings have been confirmed and explained by the the proposed ablation study, where we showed the superiority of Random Forest and LLE algorithms by performing extensive experiments (14 different classifiers and 6 dimensionality reduction methods). We also showed the capacity of LLE in the selection of important combination of features. In conclusion, our study has demonstrated that ML methods based on ensemble learning can achieve high performance in COVID-19 diagnosis from cough sound data. Our developed LLE-LLD-RF approach has shown superior performance to other approaches, while also being time-effective. Our findings emphasize the crucial need to pay more attention to the sensitivity metric in the context of COVID-19 diagnosis, and the importance of dimensionality reduction in extracting and combining the most informative patterns from cough sound data.

## GENERAL CONCLUSION

ML and DL techniques have gained significant attention in the field of medical diagnosis and healthcare. These approaches offer the ability to analyze complex datasets containing medical images, audio signals, clinical or genetic information. Within the context of respiratory diseases, such as asthma, COPD, and pneumonia, ML and DL have demonstrated a high ability in aiding healthcare professionals by identifying important patterns, features and regions of interest that are often challenging for humans to discern. However, the emergence of COVID-19 pandemic has presented a challenge for healthcare systems globally. Previously, the diagnosis of COVID-19 was primarily based on RT-PCR tests, which proved to be expensive, time-consuming, and necessitated specialized laboratory facilities. The lack of test kits and the resulting delay in obtaining results blocked efforts to contain the spread of the virus effectively. Consequently, there emerged an urgent demand for a quick and accurate diagnostic approach for COVID-19. This pressing need prompted the exploration of novel and innovative methods within the context of medical diagnosis. ML and DL techniques offer promising solutions to address the challenges posed by traditional diagnostic methods. By leveraging clinical data, X-ray images, and CT scans, researchers have endeavored to build robust diagnostic systems. However, it is important to note that clinical data alone may not suffice for precise diagnosis, and relying solely on X-ray and CT scans may not consistently yield conclusive evidence of the virus's presence. While X-ray and CT scans have demonstrated high diagnosis performance, they require physical access to the patient, which may compromise safety protocols.

Several researchers have explored the possibility of using respiratory sounds, particularly, cough to diagnose COVID-19. As previously reported, coughing is one of the most common symptoms of COVID-19. The use of cough sound data offers several advantages over other methods of diagnosis. First, it is non-invasive, which eliminates the risk of infection transmission

during sample collection. Second, can be inferred through web or mobile app, and the most important advantage is the safety of the solution, where users are supposed to use the diagnosis system remotely, and then, preventing the break of health rules imposed by authorities.

This thesis aimed to develop a quick, accurate, and safe way to diagnose COVID-19 from solely cough sound recording using ML and DL techniques. The first research question of this thesis aimed to address the feasibility of using ML-assisted diagnosis systems to achieve a performance level that is comparable to the current golden standard of RT-PCR, while the second question sought to investigate whether DL methods are necessary to develop an accurate and efficient diagnosis system for COVID-19 based on cough sound data, or if ML methods can achieve a comparable level of performance. The third research question focused on the strategies to address the issue of class imbalance which is present in the used cough sound dataset COUGHVID and also other existing datasets, and how to effectively overcome that.

In order to address the aforementioned research questions, a set of specific objectives were established and pursued throughout this thesis, which aimed to provide an in-depth theoretical background about ML, DL, dimensionality reduction, and data augmentation, specifically, the ones that were widely used in the present work. The background part included as well a concise summary of the state of the art of ML and DL in the diagnosis and detection of various respiratory diseases such as Pneumonia, Lung cancer, and OSA. The background was then followed by a critical and detailed review of the literature and state-of-the-art COVID-19 detection from cough sound using ML and DL, where 40 papers were analyzed and categorized into three main groups: ML-based approaches, DL-based approaches, and studies that compared the performance of both approaches. A comparative analysis has been done from different perspectives; ML vs. DL, feature extraction and selection techniques, and the employed data augmentation and class balancing strategies. Finally, some limitations and challenges of the discussed works were highlighted: 1) The need to have high-quality cough data which has relation to the difficulty of collecting cough samples from patients who may be under isolation or hospitalized, 2) The lack of sufficient COVID-19 samples and absence of data augmentation and class balancing techniques, which may have a significant impact on model's generalization, and 3) The inadequate consideration of the sensitivity metric, where we denoted that it has often been overlooked, ranging from 73% to 81.2% in some studies. In order to address the discussed limitations, this thesis aimed to contribute to the field with two novel approaches for COVID-19 detection from cough sound.

The first approach (Chapter 4) consists of a novel framework that combines the strengths of both CNN and LSTM along with Attention mechanism, CNN block learns the spatial features from input mel-spectrograms of cough sound signals, followed by LSTM that captures the temporal dependencies between these features, and then, additive attention mechanism is used to highlight the most informative features for the diagnosis task, while the spectral data augmentation *SpecAugment* is used to create more variability in the mel-spectrograms, allowing the model to learn more robust and generalized features. The presented contribution has been

---

compared with several baselines: CNN only, LSTM only, and hybrid CNN-LSTM, to validate and illustrate the effectiveness of each component of the architecture. Regarding achieved results, our model has been validated using 10-fold Cross-Validation and achieved a testing sensitivity of 90.93%, overall accuracy, and ROC-AUC of 91.13%. Although our approach outperformed some studies in terms of sensitivity metric, it suffers from some limitations and needs more focus on the impact of binary class transformation which may affect the model's performance due to our consideration that symptomatic samples may belong to Likely-COVID-19, sensitivity performance which needs more attention to improve the ability of the model to correctly classify Likely-COVID-19 samples. The presented study showed the impact of each of the employed DL techniques and the significant effect of spectral data augmentation in boosting the model's performance.

The second approach (Chapter 5) aimed to explore the potential of ML approaches for COVID-19 diagnosis based on cough sound data. Specifically, we developed three different ensemble learning-based frameworks that utilize the Random Forest classifier as the primary classifier: LLD-RF, Bottleneck-LLD-RF, and LLE-LLD-RF. We started our second contribution with an exploratory analysis and data engineering where we plotted four Locally Linear embeddings (EM1, EM2, EM3, and EM4) in a scatterplot matrix and denoted a sign of distinction between classes. In this respect, we put forward a hypothesis that supposes that reducing feature dimensionality may have an impact on the combination of features that can highly contribute to the classification process. To validate the hypothesis, we started by LLD-RF, the initial ML approach which involved training the model on the full LLD ComParE 2016 feature set which was extracted from cough sounds using OpenSMILE toolkit. This approach achieved a testing accuracy of 90.42%, sensitivity of 81.21%, and ROC-AUC of 90.46%. The presented approach suffers from the weakness of low sensitivity. We thought about a second approach, Bottleneck-LLD-RF, that addresses the issue by reducing dimensionality using a stacked autoencoder. This approach boosted the sensitivity from 81.21% to 86.28%. In the third proposed ML approach, named LLE-LLD-RF, we used LLE method that preserves the local data structure when moving from high-dimensional to low-dimensional data. We denote a significant enhancement for sensitivity metric, where it reached 93.89% outperforming the other two approaches. These results validate our hypothesis about dimensionality reduction and its ability to extract and combine the most important features to accurately distinguish between Likely-COVID-19 and Non-Likely-COVID-19. Regarding time complexity, we denote the model inference for the testing set of size 4951 was done in only 2.3 seconds, which made our approach time-efficient.

This work presented a significant contribution in the field of detecting COVID-19 from cough sound using ML and DL methods. The results of our contributions should give us an answer to the research questions:

**Can ML models train on cough sound data accurately distinguish between positive and negative cases of COVID-19?**

As reported in Chapters 4 and 5, the accuracy of ML models in distinguishing between positive and negative cases of COVID-19 based on cough sound data appears to be relatively better than that of some traditional diagnostic techniques, such as RT-PCR and rapid antigen tests. For instance, rapid antigen tests were reported to have a sensitivity of 60.5% and 72.1% in asymptomatic and symptomatic patients, respectively, while the sensitivity of our proposed ML approaches ranged from 81.21% to 93.89%. However, several factors can have an effect on the performance of these models, such as the quality of the data trained on and the discussed limitations. Therefore, it is challenging to conclude explicitly that our models outperform other invasive techniques. Nevertheless, our ML models may present a good alternative for testing tools, especially in cases where RT-PCR testing is not readily available, or when there is a need to quickly and accurately identify COVID-19 cases.

**Can ML methods achieve comparable performance to DL methods in developing accurate and efficient diagnosis systems for COVID-19 based on cough sound data, or is DL necessary to achieve the desired level of performance?**

Based on the achieved results of our proposed ML and DL-based frameworks, we can conclude that ML models can achieve comparable or even better performance than DL models in developing accurate and efficient diagnosis systems for COVID-19 based on cough sound data. Specifically, our LLE-LLD-RF method has explicitly outperformed the proposed DL-based framework in terms of sensitivity, with a value of 93.98% compared to 90.93%. This finding suggests that simple and low-cost ML models can achieve higher results than DL models in such cases, providing a promising alternative to expensive and complex DL approaches.

**How can the issue of class imbalance (the lack of positive samples) in COVID-19 cough sound datasets be addressed to improve the performance of ML-assisted diagnosis systems?**

Addressing the issue of class imbalance in COVID-19 cough sound datasets is a challenging task due to its significant impact on the model's performance, as the model tends to be biased towards the negative class. In our study, we applied SpecAugment to augment and add more variability to the Likely-COVID-19 class, as well as the use of SMOTE to oversample our data before training. These techniques have been shown to provide a significant improvement in our ML models' performance. However, we cannot explicitly confirm the robustness of these techniques as they require an evaluation system to check the quality of the augmented data. Therefore, further research is needed to develop more robust methods to address the class imbalance issue in COVID-19 cough sound datasets, for instance, by collecting more positive samples.

Overall, this thesis contributes to the ongoing effort to combat COVID-19 pandemic by offering

---

a promising alternative to the traditional diagnostic methods, which could help healthcare providers to quickly and accurately identify COVID-19 cases, ultimately leading to better control and prevention of the virus spread.





## BIBLIOGRAPHY

- [1] O. O. ABAYOMI-ALLI, R. DAMAŠEVIČIUS, A. QAZI, M. ADEDOYIN-OLOWE, AND S. MISRA, *Data augmentation and deep learning methods in sound classification: A systematic review*, *Electronics*, 11 (2022).
- [2] A. ABDAOUI, M. BERRIMI, M. OUSSALAH, AND A. MOUSSAOUI, *Dziribert: a pre-trained language model for the algerian dialect*, 09 2021.
- [3] B. L. Y. AGBLEY, J. LI, A. HAQ, B. COBBINAH, D. KULEVOME, P. A. AGBEFU, AND B. ELEEZA, *Wavelet-Based Cough Signal Decomposition for Multimodal Classification*, in 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2020, pp. 5–9.
- [4] M. A. A. ALBADR, S. TIUN, M. AYOB, AND F. T. AL-DHIEF, *Particle Swarm Optimization-Based Extreme Learning Machine for COVID-19 Detection*, *Cognitive Computation*, (2022).
- [5] S. ALI, Y. ZHOU, AND M. PATTERSON, *Efficient analysis of COVID-19 clinical data using machine learning models*, *Medical & Biological Engineering & Computing*, 60 (2022), pp. 1881–1896.
- [6] Y. ALIMOHAMADI, M. SEPANDI, M. TAGHDIR, AND H. HOSAMIRUDSARI, *Determine the most common clinical symptoms in COVID-19 patients: a systematic review and meta-analysis*, *J Prev Med Hyg*, 61 (2020), pp. E304–E312.
- [7] E. ALPAYDIN, *Introduction to Machine Learning*, The MIT Press, 2014.
- [8] M. ALY AND N. S. ALOTAIBI, *A novel deep learning model to detect covid-19 based on wavelet features extracted from mel-scale spectrogram of patients' cough and breathing sounds*, *Informatics in Medicine Unlocked*, 32 (2022), p. 101049.
- [9] K. ASKARI NASAB, J. MIRZAEI, A. ZALI, S. GHOLIZADEH, AND M. AKHLAGHDOUST, *Coronavirus diagnosis using cough sounds: Artificial intelligence approaches*, *Frontiers in Artificial Intelligence*, 6 (2023).
- [10] B. T. ATMAJA AND A. SASOU, *Effects of data augmentations on speech emotion recognition*, *Sensors*, 22 (2022).

## BIBLIOGRAPHY

---

- [11] D. BAHDANAU, K. CHO, AND Y. BENGIO, *Neural Machine Translation by Jointly Learning to Align and Translate*, ArXiv, 1409 (2014).
- [12] A. BERNHEIM, X. MEI, M. HUANG, Y. YANG, Z. A. FAYAD, N. ZHANG, K. DIAO, B. LIN, X. ZHU, K. LI, S. LI, H. SHAN, A. JACOBI, AND M. CHUNG, *Chest CT Findings in Coronavirus Disease-19 (COVID-19): Relationship to Duration of Infection*, *Radiology*, 295 (2020), pp. 685–691.
- [13] M. BERRIMI, S. HAMDI, R. Y. CHERIF, A. MOUSSAOUI, M. OUSSALAH, AND M. CHABANE, *COVID-19 detection from Xray and CT scans using transfer learning*, in 2021 International Conference of Women in Data Science at Taif University (WiDSTaif ), 2021, pp. 1–6.
- [14] M. BERRIMI, A. MOUSSAOUI, M. OUSSALAH, AND M. SAIDI, *Attention-based networks for analyzing inappropriate speech in arabic text*, 12 2020, pp. 1–6.
- [15] M. BERRIMI, M. OUSSALAH, A. MOUSSAOUI, AND M. SAIDI, *Attention mechanism architecture for arabic sentiment analysis*, *ACM Transactions on Asian and Low-Resource Language Information Processing*, (2022).
- [16] G. BIAU, *Analysis of a random forests model*, *J. Mach. Learn. Res.*, 13 (2012), p. 1063–1095.
- [17] B. E. BOSER, I. M. GUYON, AND V. N. VAPNIK, *A Training Algorithm for Optimal Margin Classifiers*, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, New York, NY, USA, 1992, Association for Computing Machinery, pp. 144–152.
- [18] L. BREIMAN, *Random Forests*, *Machine Learning*, 45 (2001), pp. 5–32.
- [19] C. BROWN, J. CHAUHAN, A. GRAMMENOS, J. HAN, A. HASTHANASOMBAT, D. SPATHIS, T. XIA, P. CICUTA, AND C. MASCOLO, *Exploring Automatic Diagnosis of COVID-19 from Crowdsourced Respiratory Sound Data*, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20*, New York, NY, USA, 2020, Association for Computing Machinery, pp. 3474–3484.
- [20] W. CHAN, N. JAITLEY, Q. LE, AND O. VINYALS, *Listen, attend and spell: A neural network for large vocabulary conversational speech recognition*, in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 4960–4964.
- [21] H. CHATRZARRIN, A. ARCELUS, R. GOUBRAN, AND F. KNOEFEL, *Feature extraction for the differentiation of dry and wet cough sounds*, in 2011 IEEE International Symposium on Medical Measurements and Applications, may 2011, pp. 162–166.

- [22] G. CHAUDHARI, X. JIANG, A. FAKHRY, A. HAN, J. XIAO, S. SHEN, AND A. KHANZADA, *Virufy: Global Applicability of Crowdsourced and Clinical Datasets for AI Detection of COVID-19 from Cough*, nov 2020.
- [23] N. CHAWLA, K. BOWYER, L. HALL, AND W. KEGELMEYER, *SMOTE: Synthetic Minority Over-sampling Technique*, J. Artif. Intell. Res. (JAIR), 16 (2002), pp. 321–357.
- [24] J. CHOROWSKI, D. BAHDANAU, D. SERDYUK, K. CHO, AND Y. BENGIO, *Attention-based models for speech recognition*, 2015.
- [25] N. K. CHOWDHURY, M. A. KABIR, M. M. RAHMAN, AND S. M. S. ISLAM, *Machine learning for detecting covid-19 from cough sounds: An ensemble-based mcdm method*, Computers in Biology and Medicine, 145 (2022), p. 105405.
- [26] H. COPPOCK, A. GASKELL, P. TZIRAKIS, A. BAIRD, L. JONES, AND B. SCHULLER, *End-to-end convolutional neural network enables COVID-19 detection from breath and cough audio: a pilot study*, BMJ innovations, 7 (2021), pp. 356–362.
- [27] V. CORMAN, O. LANDT, M. KAISER, R. MOLENKAMP, A. MEIJER, D. CHU, T. BLEICKER, S. BRUENINK, J. SCHNEIDER, M. SCHMIDT, D. MULDER, B. HAAGMANS, B. VEER, S. BRINK, L. WIJSMAN, G. GODERSKI, J.-L. ROMETTE, J. ELLIS, M. ZAMBON, AND C. DROSTEN, *Detection of 2019 novel coronavirus (2019-ncov) by real-time rt-pcr*, Euro-surveillance, 25 (2020).
- [28] W.-C. DAI, H.-W. ZHANG, J. YU, H.-J. XU, H. CHEN, S.-P. LUO, H. ZHANG, L.-H. LIANG, X.-L. WU, Y. LEI, AND F. LIN, *Ct imaging and differential diagnosis of covid-19*, Canadian Association of Radiologists Journal, 71 (2020), p. 084653712091303.
- [29] V. DENTAMARO, P. GIGLIO, D. IMPEDOVO, L. MORETTI, AND G. PIRLO, *Auco resnet: an end-to-end network for covid-19 pre-screening from cough and breath*, Pattern Recognition, 127 (2022), p. 108656.
- [30] M. ELGENDI, M. U. NASIR, Q. TANG, D. SMITH, J.-P. GRENIER, C. BATTE, B. SPIELER, W. D. LESLIE, C. MENON, R. R. FLETCHER, N. HOWARD, R. WARD, W. PARKER, AND S. NICOLAOU, *The effectiveness of image augmentation in deep learning networks for detecting covid-19: A geometric transformation perspective*, Frontiers in Medicine, 8 (2021).
- [31] H. ELMOAQET, M. EID, M. GLOS, M. RYALAT, AND T. PENZEL, *Deep recurrent neural networks for automatic detection of sleep apnea from single channel respiration signals*, Sensors, 20 (2020).

## BIBLIOGRAPHY

---

- [32] M. ESPOSITO, S. RAO, V. NARAYANASWAMY, AND A. SPANIAS, *Covid-19 detection using audio spectral features and machine learning*, in 2021 55th Asilomar Conference on Signals, Systems, and Computers, 2021, pp. 1146–1150.
- [33] C. FAN, *Survey of convolutional neural network*, 2016.
- [34] A. FURTADO, C. A. C. DA PURIFICAÇÃO, R. BADARÓ, AND E. G. S. NASCIMENTO, *A light deep learning algorithm for CT diagnosis of COVID-19 pneumonia*, *Diagnostics* (Basel), 12 (2022).
- [35] L. GARIBYAN AND N. AVASHIA, *Polymerase chain reaction*, *The Journal of investigative dermatology*, 133 (2013), p. e6.
- [36] P. GHOSE, M. A. UDDIN, U. ACHARJEE, AND S. SHARMIN, *Deep viewing for the identification of Covid-19 infection status from chest X-Ray image using CNN based architecture*, *Intelligent Systems with Applications*, 16 (2022).
- [37] S. GHRABLI, M. ELGENDI, AND C. MENON, *Challenges and opportunities of deep learning for cough-based covid-19 diagnosis: A scoping review*, *Diagnostics*, 12 (2022).
- [38] T. GIANNAKOPOULOS AND A. PIKRAKIS, *Chapter 4 - audio features*, in *Introduction to Audio Analysis*, T. Giannakopoulos and A. Pikrakis, eds., Academic Press, Oxford, 2014, pp. 59–103.
- [39] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep learning*. Book in preparation for MIT Press, 2016.
- [40] F. GORJI, S. SHAFIEKHANI, P. NAMDAR, S. ABDOLLAHZADE, AND S. RAFIEI, *Machine learning-based COVID-19 diagnosis by demographic characteristics and clinical data*, *Advances in respiratory medicine*, (2022).
- [41] F. GOUYON, F. PACHET, AND O. DELERUE, *On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds*, (2002).
- [42] D. GREENWOOD, T. TAVERNER, N. J. ADDERLEY, M. J. PRICE, K. GOKHALE, C. SAINSBURY, S. GALLIER, C. WELCH, E. SAPEY, D. MURRAY, H. FANNING, S. BALL, K. NIRANTHARAKUMAR, W. CROFT, AND P. MOSS, *Machine learning of COVID-19 clinical data identifies population structures with therapeutic potential*, *iScience*, 25 (2022), p. 104480.
- [43] G. GUGLIELMI, *Fast coronavirus tests: what they can and can't do*, *Nature*, 585 (2020).
- [44] S. HAMDI, A. MOUSSAOUI, M. OUSSALAH, AND M. SAIDI, *Early COVID-19 Diagnosis from Cough Sound Using Random Forest and Low-Level Descriptors*, in *The Third International Conference on Computer and Information Sciences 2021*, 2021, pp. 1–6.

- 
- [45] S. HAMDI, M. OUSSALAH, A. MOUSSAOUI, AND M. SAIDI, *Attention-based hybrid CNN-LSTM and spectral data augmentation for COVID-19 diagnosis from cough sound*, Journal of Intelligent Information Systems, (2022).
- [46] J. HAN, M. KAMBER, AND J. PEI, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2012.
- [47] J. HARVILL, Y. WANI, M. HASEGAWA-JOHNSON, N. AHUJA, D. BEISER, AND D. CHESTEK, *Classification of COVID-19 from Cough Using Autoregressive Predictive Coding Pre-training and Spectral Data Augmentation*, aug 2021.
- [48] M. M. HASAN, M. U. ISLAM, M. J. SADEQ, W.-K. FUNG, AND J. UDDIN, *Review on the evaluation and development of artificial intelligence for covid-19 containment*, Sensors, 23 (2023).
- [49] Y. HE, X. ZHENG, AND Q. MIAO, *Tfa-clstmnn: Novel convolutional network for sound-based diagnosis of covid-19*, International Journal of Wavelets, Multiresolution and Information Processing, 21 (2023), p. 2250058.
- [50] E. E.-D. HEMDAN, W. EL-SHAFAI, AND A. SAYED, *CR19: a framework for preliminary detection of COVID-19 in cough audio signals using machine learning algorithms for automated medical diagnosis applications*, Journal of Ambient Intelligence and Humanized Computing, (2022).
- [51] S. HOCHREITER, *The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6 (1998), pp. 107–116.
- [52] S. HOCHREITER AND J. SCHMIDHUBER, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–80.
- [53] J. HU, L. SHEN, S. ALBANIE, G. SUN, AND E. WU, *Squeeze-and-excitation networks*, 2019.
- [54] M. HUSAIN, A. SIMPKIN, C. GIBBONS, T. TALKAR, D. LOW, P. BONATO, S. S. GHOSH, T. QUATIERI, AND D. T. O’KEEFFE, *Artificial intelligence for detecting covid-19 with the aid of human cough, breathing and speech signals: Scoping review*, IEEE Open Journal of Engineering in Medicine and Biology, 3 (2022), pp. 235–241.
- [55] A. IMRAN, I. POSOKHOVA, H. N. QURESHI, U. MASOOD, M. S. RIAZ, K. ALI, C. N. JOHN, M. D. I. HUSSAIN, AND M. NABEEL, *AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app*, Informatics in Medicine Unlocked, 20 (2020), p. 100378.

## BIBLIOGRAPHY

---

- [56] S. IOFFE AND C. SZEGEDY, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, CoRR, abs/1502.0 (2015).
- [57] R. ISLAM, E. ABDEL-RAHEEM, AND M. TARIQUE, *Early detection of covid-19 patients using chromagram features of cough sound recordings with machine learning algorithms*, in 2021 International Conference on Microelectronics (ICM), 2021, pp. 82–85.
- [58] M. F. KABIR, T. CHEN, AND S. A. LUDWIG, *A performance analysis of dimensionality reduction algorithms in machine learning models for cancer prediction*, Healthcare Analytics, 3 (2023), p. 100125.
- [59] N. KEHTARNAVAZ, *Chapter 7 - frequency domain processing*, in Digital Signal Processing System Design (Second Edition), N. Kehtarnavaz, ed., Academic Press, Burlington, second edition ed., 2008, pp. 175–196.
- [60] I. U. KHAN, N. ASLAM, T. ANWAR, H. S. ALSAIF, S. M. B. CHROUF, N. A. ALZHRANI, F. A. ALAMOUDI, M. M. KAMALELDIN, AND K. B. AWARY, *Using a Deep Learning Model to Explore the Impact of Clinical Data on COVID-19 Diagnosis Using Chest X-ray*, 2022.
- [61] S. KHANDKER, N. H. H. NIK HASHIM, Z. DERIS, R. SHUEB, AND M. A. ISLAM, *Diagnostic accuracy of rapid antigen test kits for detecting sars-cov-2: A systematic review and meta-analysis of 17,171 suspected covid-19 patients*, Journal of Clinical Medicine, 10 (2021).
- [62] J.-M. KIM, Y.-S. CHUNG, H. JO, N. LEE, M. KIM, S. WOO, S. PARK, J. KIM, H. KIM, AND M.-G. HAN, *Identification of coronavirus isolated from a patient in korea with covid-19*, Osong Public Health and Research Perspectives, 11 (2020).
- [63] S. KIM, J.-Y. BAEK, AND S.-P. LEE, *Covid-19 detection model with acoustic features from cough sound and its application*, Applied Sciences, 13 (2023).
- [64] T. KIM, J.-W. KIM, AND K. LEE, *Detection of sleep disordered breathing severity using acoustic biomarker and machine learning techniques*, BioMedical Engineering OnLine, 17 (2018), p. 16.
- [65] D. KINGMA AND J. BA, *Adam: A Method for Stochastic Optimization*, International Conference on Learning Representations, (2014).
- [66] A. KLAPURI AND M. DAVY, *Signal Processing Methods for Music Transcription*, 01 2006.
- [67] L. KRANTHI KUMAR AND P. J. A. ALPHONSE, *COVID-19 disease diagnosis with light-weight CNN using modified MFCC and enhanced GFCC from human respiratory sounds*, The European Physical Journal Special Topics, 231 (2022), pp. 3329–3346.

- 
- [68] A. LAOUREM, C. KARA-MOHAMED, E. B. BOURENANE, AND A. HAMDI-CHERIF, *A deep learning model for CXR-based COVID-19 detection*, in 2021 International Conference on Engineering and Emerging Technologies (ICEET), 2021, pp. 1–5.
- [69] Y. LECUN, B. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD, AND L. D. JACKEL, *Backpropagation Applied to Handwritten Zip Code Recognition*, *Neural Computation*, 1 (1989), pp. 541–551.
- [70] K. K. LELLA AND A. PJA, *A literature review on covid-19 disease diagnosis from respiratory sound data*, *AIMS Bioengineering*, 8 (2021), pp. 140–153.
- [71] K. K. LELLA AND A. PJA, *Automatic diagnosis of covid-19 disease using deep convolutional neural network with multi-feature channel from respiratory sound data: Cough, voice, and breath*, *Alexandria Engineering Journal*, 61 (2022), pp. 1319–1334.
- [72] H. LI, X. CHEN, X. QIAN, H. CHEN, Z. LI, S. BHATTACHARJEE, H. ZHANG, M.-C. HUANG, AND W. XU, *An explainable covid-19 detection system based on human sounds*, *Smart Health*, 26 (2022), p. 100332.
- [73] B. LIU AND I. LANE, *Attention-based recurrent neural network models for joint intent detection and slot filling*, 2016.
- [74] G. LIU, F. LIU, J. GU, X. MAO, X. XIE, AND J. SANG, *An attention-based deep learning network for lung nodule malignancy discrimination*, *Frontiers in Neuroscience*, 16 (2023), p. 1106937.
- [75] N. LIU, M. L. CHEE, Z. X. KOH, S. L. LEOW, A. F. W. HO, D. GUO, AND M. E. H. ONG, *Utilizing machine learning dimensionality reduction for risk stratification of chest pain patients in the emergency department*, *BMC Medical Research Methodology*, 21 (2021), p. 74.
- [76] M. LOEY AND S. MIRJALILI, *Covid-19 cough sound symptoms classification from scalogram image representation using deep learning models*, *Computers in Biology and Medicine*, 139 (2021), p. 105020.
- [77] S. K. MAHANTA, D. KAUSHIK, H. VAN TRUONG, S. JAIN, AND K. GUHA, *Covid-19 diagnosis from cough acoustics using convnets and data augmentation*, in 2021 First International Conference on Advances in Computing and Future Communication Technologies (ICACFCT), 2021, pp. 33–38.
- [78] A. MANICKAM, J. JIANG, Y. ZHOU, A. SAGAR, R. SOUNDRAPANDIYAN, AND R. DINESH JACKSON SAMUEL, *Automated pneumonia detection on chest X-ray images: A deep learning approach with different optimizers and transfer learning architectures*, *Measurement*, 184 (2021), p. 109953.

## BIBLIOGRAPHY

---

- [79] W. S. McCULLOCH AND W. PITTS, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 5 (1943), pp. 115–133.
- [80] M. MELEK, *Diagnosis of covid-19 and non-covid-19 patients by classifying only a single cough sound*, Neural Computing and Applications, 33 (2021).
- [81] N. MELEK MANSHOURI, *Identifying COVID-19 by using spectral analysis of cough recordings: a distinctive classification study*, Cognitive Neurodynamics, 16 (2022), pp. 239–253.
- [82] A. MIKOŁAJCZYK AND M. GROCHOWSKI, *Data augmentation for improving deep learning in image classification problem*, 05 2018, pp. 117–122.
- [83] M. J. MINA, R. PARKER, AND D. B. LARREMORE, *Rethinking covid-19 test sensitivity — a strategy for containment*, New England Journal of Medicine, 383 (2020), p. e120. PMID: 32997903.
- [84] E. A. MOHAMMED, M. KEYHANI, A. SANATI-NEZHAD, S. H. HEJAZI, AND B. H. FAR, *An ensemble learning approach to digital corona virus preliminary screening from cough sounds*, Scientific Reports, 11 (2021), p. 15404.
- [85] M. A. MORAGHEB, A. BADIE, AND A. NOSHAD, *An effective approach for automated lung node detection using ct scans*, Journal of Biomedical Physics and Engineering, 12 (2022), pp. 377–386.
- [86] P. MOUAWAD, T. DUBNOV, AND S. DUBNOV, *Robust Detection of COVID-19 in Cough Sounds*, SN Computer Science, 2 (2021), p. 34.
- [87] A. MUGULI, L. PINTO, N. R., P. KRISHNAN, P. GHOSH, R. KUMAR, S. BHAT, S. CHETUPALLI, S. GANAPATHY, S. RAMOJI, AND V. NANDA, *DiCOVA Challenge: Dataset, Task, and Baseline System for COVID-19 Diagnosis Using Acoustics*, aug 2021.
- [88] ———, *DiCOVA Challenge: Dataset, Task, and Baseline System for COVID-19 Diagnosis Using Acoustics*, aug 2021.
- [89] U. MUHAMMAD, M. Z. HOQUE, M. OUSSALAH, A. KESKINARKAUS, T. SEPPÄNEN, AND P. SARDER, *SAM: Self-augmentation mechanism for COVID-19 detection using chest X-ray images*, Knowledge-Based Systems, 241 (2022), p. 108207.
- [90] M.-Y. NG, E. Y. P. LEE, J. YANG, F. YANG, X. LI, H. WANG, M. M.-S. LUI, C. S.-Y. LO, B. LEUNG, P.-L. KHONG, C. K.-M. HUI, K.-Y. YUEN, AND M. D. KUO, *Imaging Profile of the COVID-19 Infection: Radiologic Findings and Literature Review*, Radiology: Cardiothoracic Imaging, 2 (2020), p. e200034.



- 
- [91] L. ORLANDIC, T. TEIJEIRO, AND D. ATIENZA, *The COUGHVID crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms*, *Scientific Data*, 8 (2021), p. 156.
- [92] M. PAHAR, M. KLOPPER, R. WARREN, AND T. NIESLER, *COVID-19 cough classification using machine learning and global smartphone recordings*, *Computers in Biology and Medicine*, 135 (2021), p. 104572.
- [93] M. PAHAR, M. KLOPPER, R. WARREN, AND T. NIESLER, *Covid-19 detection in cough, breath and speech using deep transfer learning and bottleneck features*, *Computers in Biology and Medicine*, 141 (2022), p. 105153.
- [94] A. PAL AND M. SANKARASUBBU, *Pay attention to the cough: Early diagnosis of covid-19 using interpretable symptoms embeddings with cough sound signal processing*, in *Proceedings of the 36th Annual ACM Symposium on Applied Computing, SAC '21*, New York, NY, USA, 2021, Association for Computing Machinery, p. 620–628.
- [95] D. PARK, W. CHAN, Y. ZHANG, C.-C. CHIU, B. ZOPH, E. CUBUK, AND Q. LE, *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*, sep 2019.
- [96] A. PATLE AND D. S. CHOUHAN, *Sum kernel functions for classification*, in *2013 International Conference on Advances in Technology and Engineering (ICATE)*, 2013, pp. 1–9.
- [97] S. PRASAD, V. POTDAR, S. CHERIAN, P. ABRAHAM, AND A. BASU, *Transmission electron microscopy imaging of sars-cov-2*, *Indian Journal of Medical Research*, 151 (2020).
- [98] T. RAHMAN, N. IBTEHAZ, A. KHANDAKAR, M. S. A. HOSSAIN, Y. M. S. MEKKI, M. EZEDDIN, E. H. BHUIYAN, M. A. AYARI, A. TAHIR, Y. QIBLAWEY, S. MAHMUD, S. M. ZUGHAIER, T. ABBAS, S. AL-MAADEED, AND M. E. H. CHOWDHURY, *Qucoughscope: An intelligent application to detect covid-19 patients using cough and breath sounds*, *Diagnostics*, 12 (2022).
- [99] S. RAO, V. NARAYANASWAMY, M. ESPOSITO, J. J. THIAGARAJAN, AND A. SPANIAS, *COVID-19 detection using cough sound analysis and deep learning algorithms*, *Intelligent Decision Technologies*, 15 (2021), pp. 655–665.
- [100] REN, ZHAO, CHANG, YI, NEJDL, WOLFGANG, AND SCHULLER, BJÖRN W., *Learning complementary representations via attention-based ensemble learning for cough-based covid-19 recognition*, *Acta Acust.*, 6 (2022), p. 29.
- [101] S. ROWEIS AND L. SAUL, *Nonlinear dimensionality reduction by locally linear embedding*, *Science (New York, N.Y.)*, 290 (2001), pp. 2323–6.

## BIBLIOGRAPHY

---

- [102] D. E. RUMELHART, G. E. HINTON, AND R. J. WILLIAMS, *Learning representations by back-propagating errors*, *Nature*, 323 (1986), pp. 533–536.
- [103] K. SANTOSH, N. RASMUSSEN, M. MAMUN, AND S. ARYAL, *A systematic review on cough sound analysis for covid-19 diagnosis and screening: is my cough sound covid-19?*, *PeerJ Computer Science*, 8 (2022), p. e958.
- [104] R. SARMENTO AND V. COSTA, *Introduction to Linear Regression*, 01 2017.
- [105] B. SCHULLER, A. BATLINER, C. BERGLER, C. MASCOLO, J. HAN, I. LEFTER, H. KAYA, S. AMIRIPARIAN, A. BAIRD, L. STAPPEN, S. OTTL, M. GERCZUK, P. TZIRAKIS, C. BROWN, C. JAGMOHAN, A. GRAMMENOS, A. HASTHANASOMBAT, D. SPATHIS, T. XIA, AND C. KAANDORP, *The INTERSPEECH 2021 Computational Paralinguistics Challenge: COVID-19 Cough, COVID-19 Speech, Escalation & Primates*, feb 2021.
- [106] B. SCHULLER, S. STEIDL, A. BATLINER, J. EPPS, F. EYBEN, F. RINGEVAL, E. MARCHI, AND Y. ZHANG, *The INTERSPEECH 2014 computational paralinguistics challenge: cognitive & physical load*, in *Proc. Interspeech 2014*, 2014, pp. 427–431.
- [107] B. SCHULLER, S. STEIDL, A. BATLINER, J. HIRSCHBERG, J. K. BURGOON, A. BAIRD, A. C. ELKINS, Y. ZHANG, E. COUTINHO, AND K. EVANINI, *The interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language*, in *Interspeech*, 2016.
- [108] L. SETTI, F. PASSARINI, G. DE GENNARO, P. BARBIERI, M. PERRONE, M. BORELLI, J. PALMISANI, A. GILIO, P. PISCITELLI, AND A. MIANI, *Airborne transmission route of covid-19: Why 2 meters /6 feet of inter-personal distance could not be enough*, *International Journal of Environmental Research and Public Health*, 17 (2020), p. 2932.
- [109] I. SHAFI, S. DIN, A. KHAN, I. D. L. T. DÍEZ, R. D. J. P. CASANOVA, K. T. PIFARRE, AND I. ASHRAF, *An effective method for lung cancer diagnosis from ct scan using deep learning-based support vector network*, *Cancers*, 14 (2022).
- [110] A. I. SHARAF, *Sleep apnea detection using wavelet scattering transformation and random forest classifier*, *Entropy*, 25 (2023).
- [111] M. SHARMA, D. KUMBHANI, J. TIWARI, T. S. KUMAR, AND U. R. ACHARYA, *Automated detection of obstructive sleep apnea in more than 8000 subjects using frequency optimized orthogonal wavelet filter bank with respiratory and oximetry signals*, *Computers in Biology and Medicine*, 144 (2022), p. 105364.
- [112] N. SHARMA, P. KRISHNAN, R. KUMAR, S. RAMOJI, S. R. CHETUPALLI, N. R., P. K. GHOSH, AND S. GANAPATHY, *Coswara — A Database of Breathing, Cough, and Voice Sounds for COVID-19 Diagnosis*, in *Proc. Interspeech 2020*, 2020, pp. 4811–4815.

- [113] A. SHETA, H. TURABIEH, T. THAHER, J. TOO, M. MAFARJA, M. S. HOSSAIN, AND S. R. SURANI, *Diagnosis of obstructive sleep apnea from ecg signals using machine learning and deep learning classifiers*, Applied Sciences, 11 (2021).
- [114] I. SHIRI, M. SOROURI, P. GERAMIFAR, M. NAZARI, M. ABDOLLAHI, Y. SALIMI, B. KHOSRAVI, D. ASKARI, L. AGHAGHAZVINI, G. HAJIANFAR, A. KASAEIAN, H. ABDOLLAHI, H. ARABI, A. RAHMIM, A. R. RADMARD, AND H. ZAIDI, *Machine learning-based prognostic modeling using clinical data and quantitative radiomic features from chest CT images in COVID-19 patients*, Computers in Biology and Medicine, 132 (2021), p. 104304.
- [115] E. SHOWKATIAN, M. SALEHI, H. GHAFFARI, R. REIAZI, AND N. SADIGHI, *Deep learning-based automatic detection of tuberculosis disease in chest X-ray images*, Polish Journal of Radiology, 87 (2022), pp. 118–124.
- [116] H. SKANDER, A. MOUSSAOUI, M. OUSSALAH, AND M. SAIDI, *Gender Identification from Arabic Speech Using Machine Learning*, 2020, pp. 149–162.
- [117] I. SÖDERGREN, M. P. NODEH, P. C. CHHIPA, K. NIKOLAIDOU, AND G. KOVÁCS, *Detecting COVID-19 from Audio Recording of Coughs Using Random Forests and Support Vector Machines*, pp. 916–920.
- [118] F. Z. SOLAK, *Identification of covid-19 from cough sounds using non-linear analysis and machine learning*, Avrupa Bilim ve Teknoloji Dergisi, (2021), pp. 710 – 716.
- [119] M. SOLTANIAN AND K. BORNA, *Covid-19 recognition from cough sounds using lightweight separable-quadratic convolutional network*, Biomedical Signal Processing and Control, 72 (2022), p. 103333.
- [120] Q. SONG, L. ZHAO, X. LUO, AND X. DOU, *Using deep learning for classification of lung nodules on computed tomography images*, Journal of Healthcare Engineering, 2017 (2017), pp. 1–7.
- [121] S. S. STEVENS, J. VOLKMANN, AND E. B. NEWMAN, *A scale for the measurement of the psychological magnitude pitch*, The Journal of the Acoustical Society of America, 8 (1937), pp. 185–190.
- [122] I. SUTSKEVER, O. VINYALS, AND Q. V. LE, *Sequence to sequence learning with neural networks*, 2014.
- [123] K. SUZUKI, *Artificial Neural Networks: Methodological Advances and Biomedical Applications*, IntechOpen, 2011.

## BIBLIOGRAPHY

---

- [124] K. SUZUKI, *ARTIFICIAL NEURAL NETWORKS – ARCHITECTURES AND APPLICATIONS*, 01 2013.
- [125] S. SWAMINATHAN, K. QIRKO, T. SMITH, E. CORCORAN, N. G. WYSHAM, G. BAZAZ, G. KAPPEL, AND A. N. GERBER, *A machine learning approach to triaging patients with chronic obstructive pulmonary disease*, PLOS ONE, 12 (2017), pp. 1–21.
- [126] A. TAHAMTAN AND A. ARDEBILI, *Real-time RT-PCR in COVID-19 detection: issues affecting the results*, Expert Review of Molecular Diagnostics, 20 (2020), pp. 453–454.
- [127] A. G. TAYLOR, C. MIELKE, AND J. MONGAN, *Automated detection of moderate and large pneumothorax on frontal chest x-rays using deep convolutional neural networks: A retrospective study*, PLOS Medicine, 15 (2018), pp. 1–15.
- [128] A. TENA, F. CLARIÀ, AND F. SOLSONA, *Automated detection of COVID-19 cough*, Biomedical Signal Processing and Control, 71 (2022), p. 103175.
- [129] W. THORPE, M. KURVER, G. KING, AND C. SALOME, *Acoustic analysis of cough*, in The Seventh Australian and New Zealand Intelligent Information Systems Conference, 2001, nov 2001, pp. 391–394.
- [130] S. ULUKAYA, A. A. SARICA, O. ERDEM, AND A. KARAALI, *MSCCov19Net: multi-branch deep learning model for COVID-19 detection from cough sounds*, Medical & Biological Engineering & Computing, (2023).
- [131] J. UTHOFF, M. STEPHENS, J. NEWELL, JR, E. HOFFMAN, J. LARSON, N. KOEHN, F. DE STEFANO, C. LUSK, A. WENZLAFF, D. WATZA, C. NESLUND-DUDAS, L. CARR, D. LYNCH, A. SCHWARTZ, AND J. (DE RYK) SIEREN, *Machine learning approach for distinguishing malignant and benign lung nodules utilizing standardized perinodular parenchymal features from ct*, Medical Physics, 46 (2019).
- [132] J. J. VALDÉS, P. XI, M. COHEN-MCFARLANE, B. WALLACE, R. GOUBRAN, AND F. KNOEFEL, *Analysis of cough sound measurements including covid-19 positive cases: A machine learning characterization*, in 2021 IEEE International Symposium on Medical Measurements and Applications (MeMeA), 2021, pp. 1–6.
- [133] N. VAN DOREMALEN, T. BUSHMAKER, D. MORRIS, M. HOLBROOK, A. GAMBLE, B. WILLIAMSON, A. TAMIN, J. HARCOURT, N. THORNBURG, S. GERBER, J. LLOYD-SMITH, E. WIT, AND V. MUNSTER, *Aerosol and surface stability of sars-cov-2 as compared with sars-cov-1*, New England Journal of Medicine, 382 (2020).
- [134] D. S. VIJAYAKUMAR AND M. SNEHA, *Low cost covid-19 preliminary diagnosis utilizing cough samples and keenly intellectual deep learning approaches*, Alexandria Engineering Journal, 60 (2021), pp. 549–557.

- [135] J. VRINDAVANAM, R. SRINATH, H. H. SHANKAR, AND G. NAGESH, *Machine Learning based COVID-19 Cough Classification Models - A Comparative Analysis*, in 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 420–426.
- [136] S. HAMDI, A. MOUSSAOUI, M. OUSSALAH, AND M. SAIDI, *Locally Linear Embedding and Ensemble learning for COVID-19 Screening from Cough Sound*. 2023.
- [137] S. HAMDI, A. MOUSSAOUI, M. OUSSALAH, AND M. SAIDI, *Autoencoders and ensemble-based solution for covid-19 diagnosis from cough sound*, in Modelling and Implementation of Complex Systems, S. Chikhi, G. Diaz-Descalzo, A. Amine, A. Chaoui, D. E. Saidouni, and M. K. Kholadi, eds., Cham, 2023, Springer International Publishing, pp. 279–291.
- [138] A. LAOUAREM, C. KARA-MOHAMED, E. B. BOURENANE, AND A. HAMDI-CHERIF, *COVID-19 detection using CXR images: simple CNN vs. transfer learning*, in 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET), 2021, pp. 1–6.
- [139] M. BERRIMI, M. OUSSALAH, A. MOUSSAOUI, AND M. SAIDI, *A Comparative Study of Effective Approaches for Arabic Text Classification*, SSRN Electronic Journal, (2023).
- [140] C. WALL, L. ZHANG, Y. YU, A. KUMAR, AND R. GAO, *A deep ensemble neural network with attention mechanisms for lung abnormality classification using audio inputs*, Sensors, 22 (2022).
- [141] Y. WANG, W. DU, C. CAI, AND Y. XU, *Explaining the attention mechanism of end-to-end speech recognition using decision trees*, 2021.
- [142] F. WENINGER, F. EYBEN, B. SCHULLER, M. MORTILLARO, AND K. SCHERER, *On the acoustics of emotion in audio: What speech, music, and sound have in common*, Frontiers in Psychology, 4 (2013).
- [143] W. H. O. (WHO), *Coronavirus disease (covid-19) pandemic*, 2023.
- [144] S. WOO, J. PARK, J.-Y. LEE, AND I. S. KWEON, *Cbam: Convolutional block attention module*, in Computer Vision – ECCV 2018, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., Cham, 2018, Springer International Publishing, pp. 3–19.
- [145] WORLDOMETERS, *Covid-19 coronavirus pandemic*, 2023.
- [146] T. XIA, D. SPATHIS, B. CHLOË, J. CH, A. GRAMMENOS, J. HAN, A. HASTHANASOMBAT, E. BONDAREVA, T. DANG, A. FLOTO, P. CICUTA, AND C. MASCOLO, *COVID-19 sounds: A large-scale audio dataset for digital COVID-19 detection*, in Thirty-fifth Conference

## BIBLIOGRAPHY

---

- on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.
- [147] H. XUE AND F. D. SALIM, *Exploring self-supervised representation ensembles for covid-19 cough classification*, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '21, New York, NY, USA, 2021, Association for Computing Machinery, p. 1944–1952.
- [148] L. YANG, R.-Y. ZHANG, L. LI, AND X. XIE, *Simam: A simple, parameter-free attention module for convolutional neural networks*, in International Conference on Machine Learning, 2021.
- [149] Z. YANG, L. ZHU, Y. WU, AND Y. YANG, *Gated channel transformation for visual recognition*, 06 2020, pp. 11791–11800.
- [150] W. YU, L. KANG, Z. GUO, J. LIU, M. LIU, AND W. LIANG, *Incubation period of covid-19 caused by unique sars-cov-2 strains: A systematic review and meta-analysis*, JAMA Network Open, 5 (2022), p. e2228008.
- [151] J. ZHANG, Z. TANG, J. GAO, L. LIN, Z. LIU, H. WU, F. LIU, AND R. YAO, *Automatic Detection of Obstructive Sleep Apnea Events Using a Deep CNN-LSTM Model*, Computational Intelligence and Neuroscience, 2021 (2021), p. 5594733.
- [152] Y. ZOABI, S. DERI-ROZOV, AND N. SHOMRON, *Machine learning-based prediction of COVID-19 diagnosis based on symptoms*, npj Digital Medicine, 4 (2021), p. 3.

## APPENDIX: DIAGNOSIS MOBILE APPLICATION

### Introduction

The diagnosis mobile application aims to provide a safe and remote solution for efficient COVID-19 screening. Recognizing the importance of an end-to-end process, the application enables users to easily submit their cough recording to the diagnosis system for analysis and classification. Various solutions can be implemented, such as embed system and web application. However, with the widespread availability of smartphones, a mobile implementation becomes the ideal solution, as it allows for easy access to the system. By making the application available on different app stores, it ensures widespread accessibility, making the diagnosis process quick and straightforward for individuals.

### Technical Implementation Details

We opted to use the Flutter framework<sup>1</sup>, which allowed us to develop Android, Web, Windows, Linux, macOS, Fuchsia and iOS versions of the application using a single codebase. In our technical implementation and building configuration, we targeted Android and iOS. Flutter provides a rich set of pre-built UI components and efficient performance, enabling us to create a seamless and visually appealing user interface for both platforms. As for the backend, we utilized the Flask micro web framework<sup>2</sup>. Flask is a lightweight and flexible framework that offers simplicity and ease of use in building web applications. Its extensive ecosystem and powerful features, such as routing and request handling, allowed us to efficiently develop the backend functionality of the diagnosis application. Flask's scalability and compatibility with various libraries and extensions provided us with a solid foundation to handle the data processing and communication between the frontend and backend components.

### Systematic Overview of the Developed System

We present a comprehensive illustration of the general workflow of the application. Figure 1 depicts the step-by-step process starting from the user's interaction with the app, the request

---

<sup>1</sup><https://flutter.dev>

<sup>2</sup><https://flask.palletsprojects.com/>

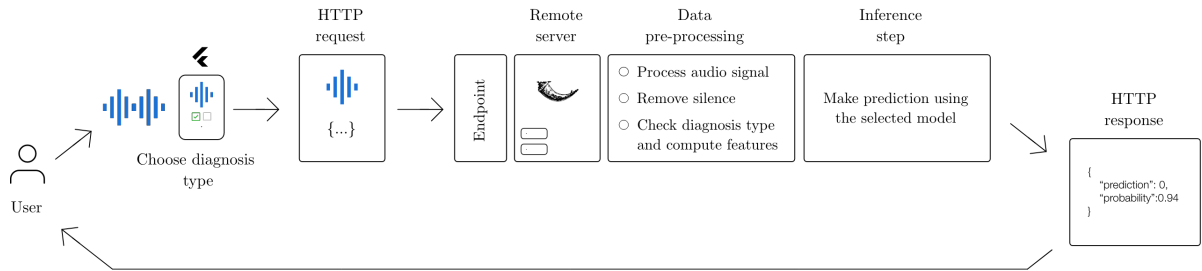


Figure 1: Overview of the proposed end-to-end diagnosis system

being sent to the backend system, the pre-processing steps performed on the data, and finally, the response provided to the user.

## Diagnosis Flow and Use Cases

The diagnosis flow of the mobile application is designed to provide users with a simple and user-friendly experience for COVID-19 screening. The process begins as the user opens the app, and the home screen (see Figure 2 (a)) greets them without the need for any registration or login step. To initiate the screening process, the user simply taps the **Get Started** button (see Figure 2 (b)).



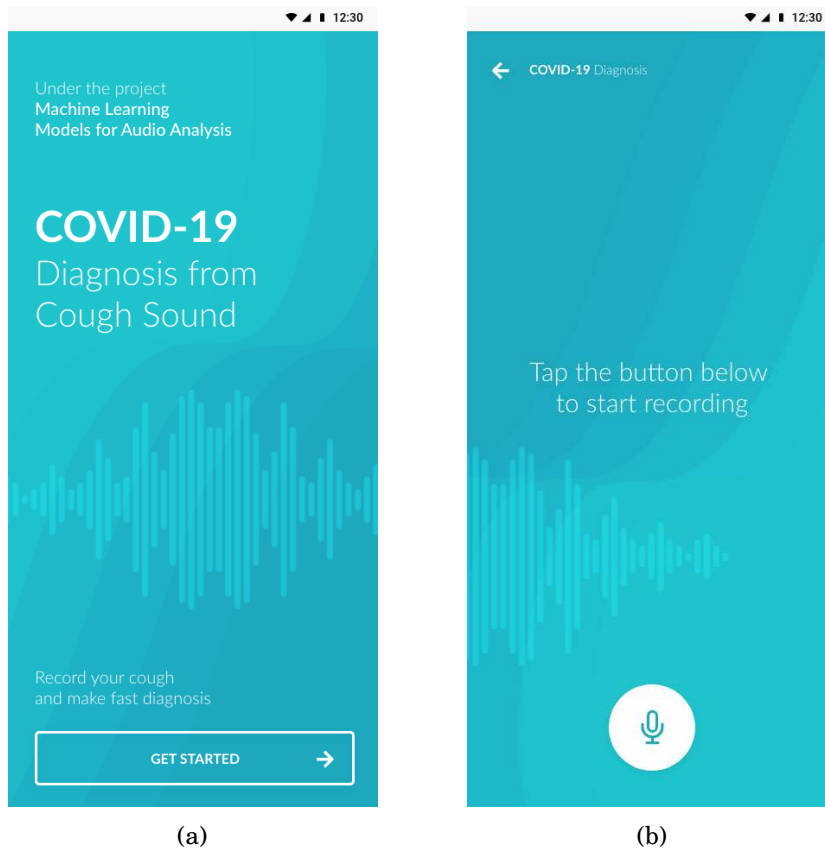


Figure 2: Mobile app home and recording screens

Upon tapping the microphone button, the user can start recording its cough sound (Figure 3 (a)), which is the key component of the system. Once the recording is complete, the user can tap the stop button. If the audio recording is successful, a new screen (Figure 3 (b)) immediately appears, informing the user that their recording is ready for analysis.

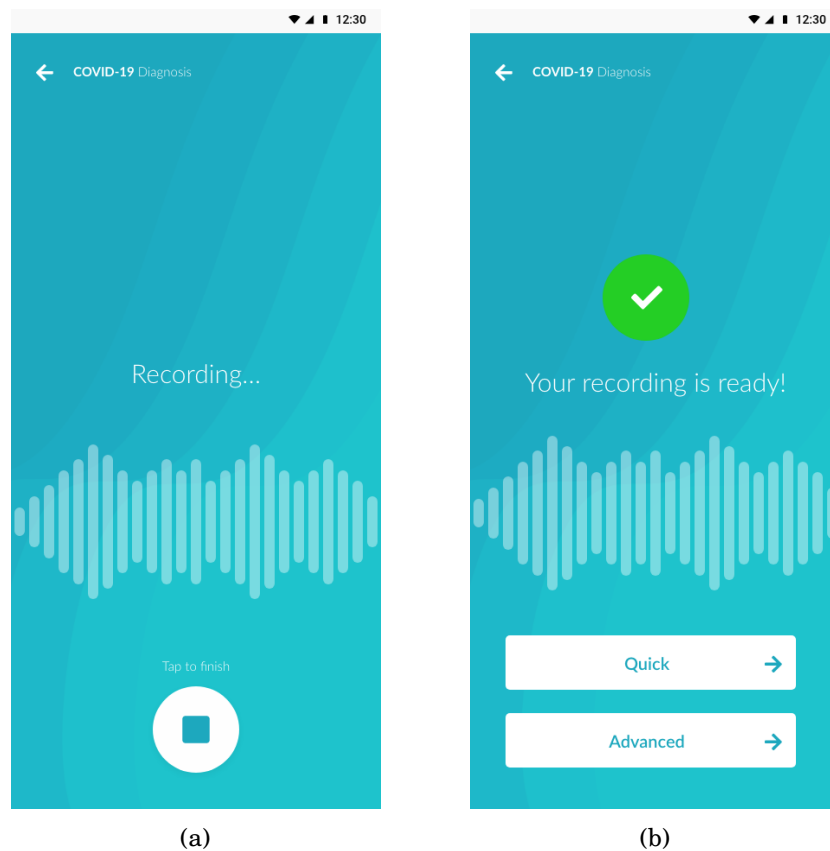


Figure 3: Recording in progress and ready screens

At this stage, the user is presented with two options: **Quick** and **Advanced** (see Figure 3 (b)). The **Quick** option triggers the LLE-LLD-RF. On the other hand, the **Advanced** option initiates the attention-based hybrid CNN-LSTM. After selecting one of the options, the user will be presented by progress indicator screen as illustrated in Figure 4 (a).

In the event of a network error, a screen indicating the error will appear, providing the user with the option to relaunch the request and try again as shown in Figure 4 (b). Similarly, if there is an error related to the silence removal process, such as an empty recording after removing silence, a dedicated screen will appear, allowing the user to restart the recording from the beginning and relaunch the request as illustrated in Figure 4 (c).

Once the analysis is complete, the app displays a screen with the model's prediction and the associated probability, providing the user with vital information about their diagnosis. This screen includes details such as the date, time, and the type of diagnosis performed (Quick or Advanced). Figures 5 (a) and 5 (b) exemplified Likely-COVID-19 and Non-Likely-COVID-19 diagnosis results, respectively.

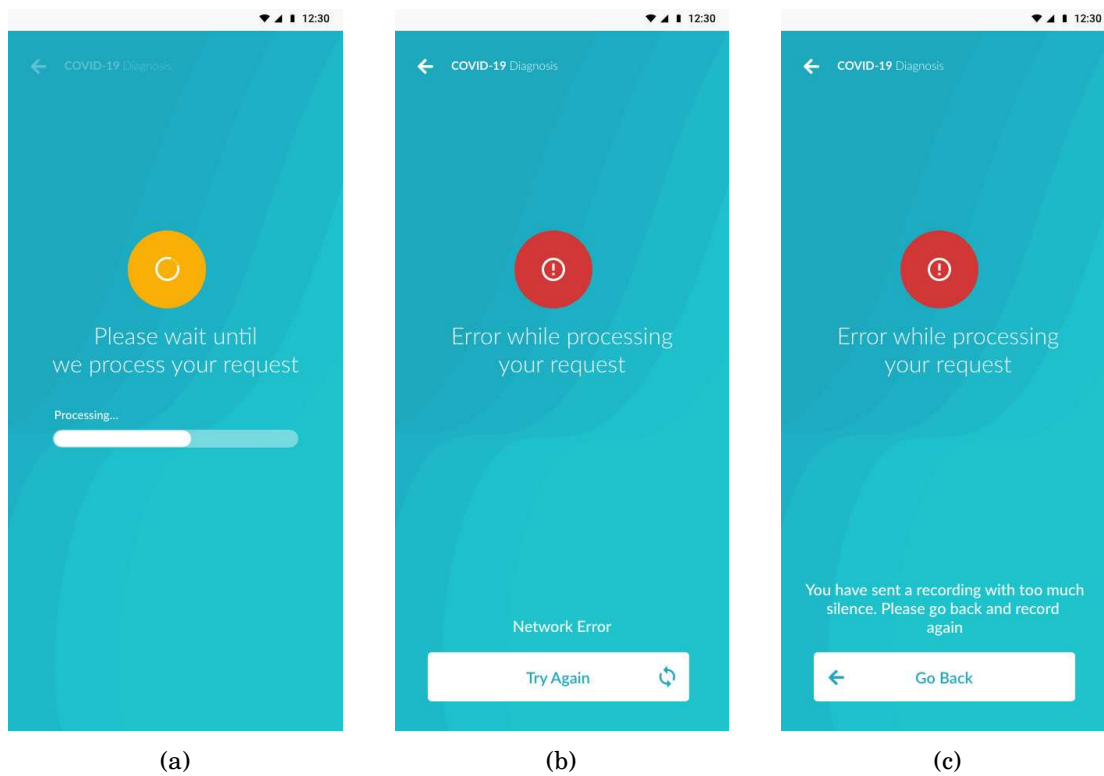


Figure 4: Request processing and error screens based on the server's response

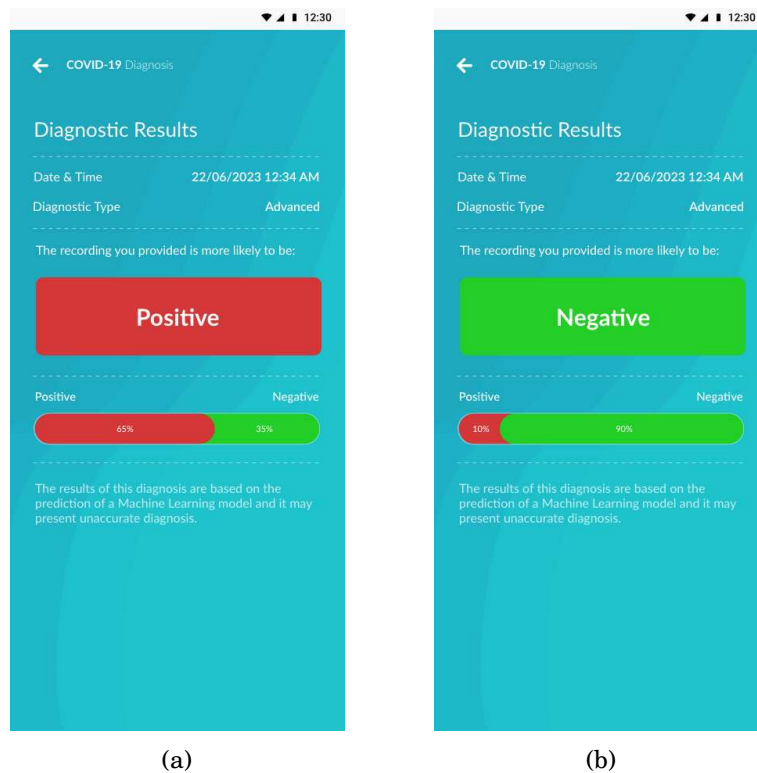


Figure 5: Model's prediction response with Likely-COVID-19 and Non-Likely-COVID-19