SAPIENZA
UNIVERSITÀ DI ROMA

Analysis and Classification of 12-channel ECG signal using Edge Computing

Facoltà di Ingegneria Dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Data Science

Candidate

Hafiz Muhammad Hassan
ID number 1873829

Thesis Advisor

Prof. Ioannis Chatzigiannakis

Academic Year 2021/2022

Thesis defended on 15 July 2022
in front of a Board of Examiners composed by:

Prof. Cuomo Francesca (Presidente) (chairman)
Prof. Anagnostopoulos Aristidis
Prof. Brutti Pierpaolo
Prof. Chatzigiannakis Ioannis
Prof. Crespi Mattia
Prof. Di Lorenzo Paolo
Prof. Navigli Roberto

Analysis and Classification of 12-channel ECG signal using Edge Computing
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: hassan.1873829@studenti.uniroma1.it

Acknowledgments

I want to convey my gratitude to professor Ioannis Chatzigiannakis, my supervisor, for guiding me through thesis project with endeavour and commitment. It was a pleasant and educational journey to work under his leadership. I also extend this acknowledgement to Daniel and Lorella, my colleagues in this thesis, since they made spectacular contributions to this project.

I want to express my gratitude to my girlfriend Hasna, who supported me while developing this thesis with love and patience.

I also want to thank all faculty of data science program because they provided me with that knowledge that made me complete this thesis project. And last but not least, I would also like to thank family and friend for their support and contributions while studying in a foreign country.

Contents

Chapter 1

Introduction

The design of advanced health-monitoring systems has always been a topic of active research. During the past decades, numerous portable devices have been introduced for the early detection and diagnosis of heart failure, since it is a common, costly, disabling, and deadly syndrome. Advanced heart-monitoring devices are capable of providing reliable, accurate heart monitoring and are able to detect sporadic events during periods of time when things would otherwise be unclear. Functioning as a black box, these devices have the potential to enhance the decision-making process, by notifying doctors when patients need readmission and in this way constituting an effective alternative capable of saving patients' lives. [1].

Research on artificial intelligence (AI), and particularly the advances in machine learning (ML) and deep learning (DL) have led to disruptive innovations in radiology, pathology, genomics and other fields. Modern DL models feature millions of parameters that need to be learned from sufficiently large curated data sets in order to achieve clinical-grade accuracy, while being safe, fair, equitable and generalising well to unseen data. [2]

Data storage and model training occur on these models usually on high-performance cloud servers in conventional centralized ML approaches. Multiple edge nodes also collaborate with the remote cloud to perform large distributed tasks that include both local processing and remote coordination and execution. Traditional model of cloud computing is unsuitable for applications that demand low latency, so as a result a new model of computation termed edge computing has sprung forth. Edge computing is primarily concerned with transmitting data among the devices at the edge, closer to where user applications are located, rather than to a centralized server. Edge node (or edge client, edge device) is usually the resource-constraint device that end user uses and it is geographically close to the nearest edge server, who has abundant computing resources and high bandwidth communicating with end nodes. When the edge server requires more computing power, it will connect to the cloud server. [3].

Federated Learning (FL) is a concept developed by Google researchers in 2016, as a promising solution for addressing the issues of communication costs, data privacy, and legalization. An FL approach is a distributed ML approach where models are trained on edge devices, organizations, or individuals under centralized control without sharing their local datasets. This ensures the privacy of data during the training process. An edge server or cloud server periodically gathers the trained parameters to create and update a better and more accurate model, which is sent back to the edge devices for local training. [4]

As in case of health data, which has privacy concerns, FL allows collaborative insights without transferring patient data outside of the institutions' firewalls. Instead, each participating institution's ML process takes place locally, with only model weights being share. Recent research has shown that models trained by FL can achieve performance levels comparable to ones trained on centrally hosted data sets and superior to models that only see isolated single-institutional data. [2]

1.1 Document's outline

The present thesis document is organized into six chapters.

The first chapter (1) introduces the reasons behind the need for a machine learning solution to classify the 12-leads, encouraging the introduction of Edge Computing architecture based on Federated Learning. The second chapter (2) describes the essential background to understand the cardiological basics. It contains the human heart nature and all the concepts associated with ECG monitoring, including the most common arrhythmia types.

The third chapter (3 looking at related work and understanding of dataset employed. In the fourth chapter (4), the analytical methodologies and tools are introduced and explained in detail.

The fifth chapter (5) it is exposed the results of analyzing the proposed data under both the Centralized environment and Federated Learning environment. Also, how we can convert any model to TensorFlow Lite and utilize them in either mobile devices(edge node) or any type of edge node like IoT devices. Finally, the last chapter (6) is dedicated to the overall conclusions achieved by the research and to possible future developments.

Chapter 2

Cardiological fundamentals

[5] "An electrocardiogram (ECG) is a measure of how the electrical activity of the heart changes over time, as action potentials within each myocyte propagate throughout the heart as a whole during each cardiac cycle. In other words, the ECG is the recording of the cumulative signals produced by populations of cells eliciting changes in their membrane potentials at a given point in time. The ECG provides specific waveforms of electrical differences when the atria and ventricles depolarize and repolarize."

2.1 The human heart nature

For an ECG, the human body can be thought of as a large volume conductor. It is made up of tissues and a conductive media in which the heart is suspended. The heart contracts during the cardiac cycle in response to coordinated action potentials travelling through the chambers of the heart. One section of the heart tissue is depolarized, while another is at rest or polarized, as is usual.

The intensity of the voltages observed is determined by the electrodes' orientation concerning the dipole ends. The signal amplitudes are proportional to the mass of tissue used to create the dipole at any particular time. Electrodes are typically placed on the skin's surface to detect the voltages of these electrical fields, giving rise to the ECG [5].

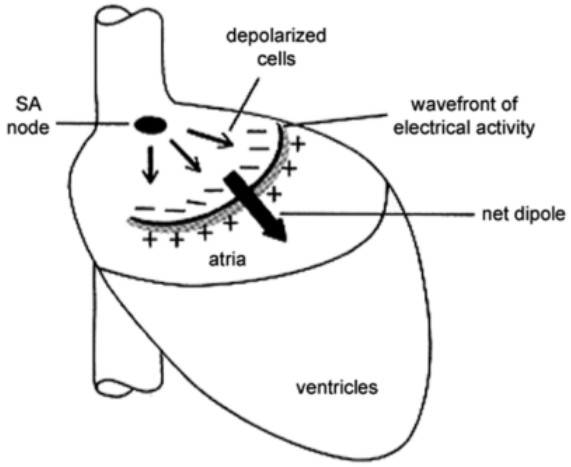


Figure 2.1. After conduction begins at the sinoatrial node, cells in the atria begin to depolarize. This creates an electrical wavefront that moves down toward the ventricles, with polarized cells at the front. The separation of charge results in a dipole across the heart (the large black arrow shows its direction) [5].

2.2 ECG's history

The discovery of intrinsic electrical activity within the heart dates back to the 1840s. Carlo Matteucci, an Italian physicist, was the first to discover that each heartbeat is accompanied by an electrical current in 1842. Emil DuBois-Reymond, a German scientist, published the first action potential associated with muscular contraction not long after. In 1856, Rudolph von Koelliker and Heinrich Miller used a galvanometer to record the first cardiac action potential. Following that, Augustus D. Waller recorded the first human ECG after Gabriel Lippmann invented the capillary electrometer in the early 1870s. That first device is shown in Figure 2.2.

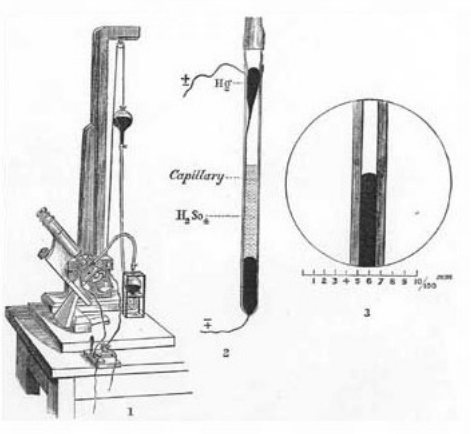


Figure 2.2. Lippmann electrometer

Willem Einthoven's creation of the string galvanometer in 1901 was a key milestone in cardiac electrocardiography. The next year, he published the first ECG using his string galvanometer. Einthoven's string galvanometer consisted of a huge electromagnet with a thin silver-coated string stretched across it; electric currents passing through the thread caused the string to move from side to side in the electromagnet's magnetic field.

Einthoven made yet another significant addition to cardiac electrophysiology in 1912, when he discovered a mathematical link between the direction and size of the deflections recorded by the three limb leads. Einthoven's triangle is the name for this hypothesis. Before Frank Wilson described unipolar leads and the precordial lead configuration, the typical three-limb leads were used for three decades. The traditional Einthoven limb leads, as well as the precordial and unipolar limb, leads based on Wilson's work, make up the 12-lead ECG layout now in use.

This instrument was initially manufactured in 1905 by the Cambridge Scientific Instrument Company in London. Electrical impulses were sent from a hospital over a mile away to Einthoven's laboratory via a telephone cable. Bedside machines, on the other hand, were not available until the 1920s. The Sanborn Company produced a smaller version of the unit in 1935 that weighed only about 25 pounds.

Figure 2.3. Holter-Edan ECG device

With Norman Jeff Holter's invention of the Holter monitor in 1949, the use of ECG in a nonclinical context became viable. The first iteration of this device was a 75-pound backpack that could record the ECG continually and send the signals via radio. The size of subsequent iterations of such devices has been drastically decreased, and the signal is now recorded digitally. Miniaturized devices now allow

patients to be monitored for longer periods (typically 24 hours) to aid in the diagnosis of any rhythm or ischemic heart disease concerns. One of the latest versions of the ECG is the one appearing in Figure 2.3.

2.3 The ECG Waveform

Signals of voltage versus time are created during the recording of an ECG, which are generally shown in millivolts (mV) vs seconds. Figure 2.4 depicts a typical Lead II ECG waveform. The negative electrode was placed on the right wrist and the positive electrode on the left ankle for this Lead II ECG recording. As a result, a series of peaks and waves can be seen, each of which corresponds to ventricular or atrial depolarization and repolarization, with each segment of the signal indicating a separate event in the cardiac cycle.

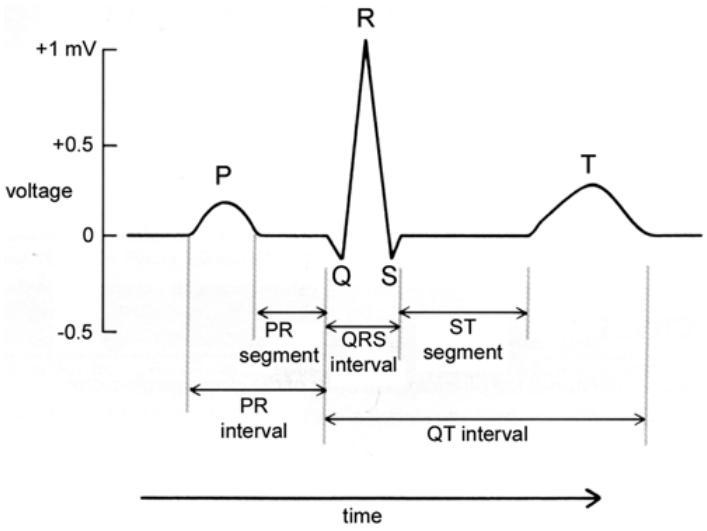


Figure 2.4. A typical ECG waveform for one cardiac cycle, measured from the Lead II position [5].

Three principal waveforms are recorded by the ECG (2.4):

- The P-wave
- QRS complex
- T-wave.

The P-wave is created by depolarization of the atria, the QRS by depolarization of the ventricles, and the T-wave by repolarisation of the ventricles. In most people, these waveforms occur in a repeating rhythm called sinus rhythm, so called because

it originates in the sinus node. In some people, a fourth waveform (not shown in the previous image) called a U-wave can be seen. This is usually seen at slower heart rates. The significance of the U-wave remains uncertain. Some authors think that it represents the late stages of ventricular repolarisation, while others describe it as a post-repolarisation phenomenon. U-wave abnormalities have been described in various disease states including ischaemic heart disease [6].

The depolarization of the sinoatrial node, which is positioned within the right atrium, starts the typical cardiac cycle. A conventional ECG will not detect this early firing because the node does not have enough cells to provide a measurable electrical potential. The right and left ventricles continue to depolarize after the P wave, resulting in the recordable QRS complex, which lasts about 100 milliseconds. The Q-wave is the initial negative deflection (if present), the R-wave is the largest positive deflection, and the S-wave is the smallest positive deflection [7].

The T-wave is usually the last potential in a cardiac cycle, followed by the P-wave of the next cycle, and so on. The ECG signal returns to baseline near the conclusion of ventricular contraction, and the ventricles repolarize after contraction. Atrial contractions have stopped and the atria are repolarizing at the same time as the QRS complex. Because the effects of this widespread atrial repolarization are obscured by the much larger volume of tissue engaged in ventricular depolarization, it is not generally detectable in an ECG [5].

2.4 The 12-leads ECG

An ECG lead is a recording of the heart's electrical activity as seen from one side. As a result, when we take a 12-lead ECG, we're recording cardiac electrical activity from 12 different angles [7]. Assume you're visiting a historic structure and taking images of it. If you snap 12 photos from different angles around the structure, each one will depict a distinct element, such as the front, sides, and back. They work together to provide a three-dimensional record of the structure's shape and appearance. Similarly, a 12-lead ECG creates a three-dimensional depiction of the heart's electrical activity.

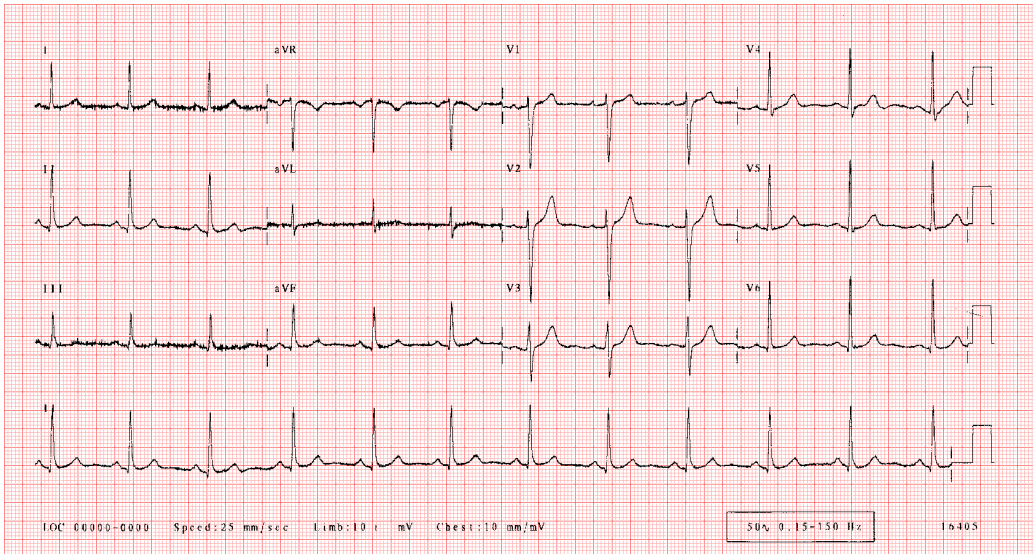


Figure 2.5. 12-leads normal ECG

Multiple images of the heart's electrical activity can be recorded depending on the type of machine utilized and the number of electrodes inserted. The usage of 12-lead ECG devices is common among healthcare professionals. Twelve separate electrical images of the heart are measured and recorded via a 12-lead ECG (Figure 2.5). In other words, it records the electrical activity of the heart as observed from 12 various angles. For example, Lead II monitors electrical activity as observed from the heart's inferior (diaphragmatic) surface. This lead is frequently used to measure heart rate [8].

2.5 Arrhythmias Types

Analyzing arrhythmias is a difficult undertaking since every person on the planet has a unique ECG that differs from everyone else's, and one person's ECG can change dramatically from one second to the next. Memorizing some of the most common ECG patterns and attempting to recognize them in the future is insufficient. Pattern identification is a popular yet unintentional way of approaching arrhythmias via ECG analysis ([9]). Often Arrhythmias are divided into two global categories:

1. **Rhythmic**: Determined as a sequence of uneven beats
2. **Morphological**: Made of abnormal single beat

The work presented in the current thesis is focused on the first type of classification. Those arrhythmias have a categorization provided by SNOMED CT. The latter is the world's most complete and precise terminology package, with widespread

acceptance around the world. It provides a common language for clinical IT systems, making data exchange between them easier, safer, and more accurate. It covers everything from processes and symptoms to clinical measurements, diagnosis, and drugs, and it's all in one place.

For the Physionet 2020 challenge only considered around 27 arrhythmias, which are the most frequently found in the ECG analysis. Nevertheless, in the following paragraphs, there is a deep explanation of all the possible arrhythmias. To see the complete list of categories it is possible to read the GitHub cited in [10].

Arrhythmias are often divided into groups based on where the rhythm is initiated by the pacemaker. The following are the most prevalent sites, and consequently the primary arrhythmia categories:

1. Sinus
2. Atrial
3. Junctional
4. Ventricular
5. AV Blocks

2.5.1 Sinus

It is necessary to comprehend the 'benchmark' rhythm, or hemodynamically perfect rhythm, which is referred to as **Normal Sinus Rhythm** and sometimes abbreviated to **NSR**, in order to assess cardiac rhythms (Figure 2.6). The following features must be present in order for a rhythm to be classified as Normal Sinus Rhythm:

Characteristic	**Status**
Rhythm	Regular
Rate	60-100/minute
p waves	Present, upright, symmetrical, one before every QRS
pri	.12-.20 seconds
QRS	.06-.10 seconds

Table 2.1. Characteristics of Normal Sinus Rhythm

Figure 2.6. Normal Sinus Rhythm ([9])

Sinus Bradycardia (SB): When a patient's heart rate falls below 60 beats per minute, they are said to be bradycardic. Slow heart rates can be seen in fit and active people who are usually asymptomatic. When a patient's heart rate falls below 60 beats per minute, critical care nurses must be ready to assess for decreasing cardiac output right away.

Characteristic	Status
Rhythm	Regular
Rate	< 60/minute
p waves	Present, upright, symmetrical, one before every QRS
pri	.12-.20 seconds
QRS	.06-.10 seconds

Table 2.2. Characteristics of Sinus Bradycardia

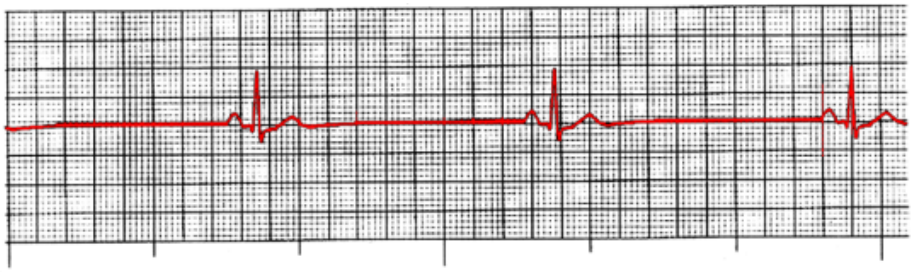


Figure 2.7. Sinus Bradycardia ([9])

Sinus Tachycardia (STach): When a patient's heart rate exceeds 100 beats per minute, they are labelled tachycardic, though most people don't notice symptoms until their heart rate exceeds 150 beats per minute. At this point, a critical care nurse should look for signs and symptoms of decreased cardiac output (such as hypotension or a loss of consciousness).

Characteristic	Status
Rhythm	Regular
Rate	>100/minute
p waves	Present, upright, symmetrical, one before every QRS
pri	.12-.20 seconds
QRS	.06-.10 seconds

Table 2.3. Characteristics of Sinus Tachycardia

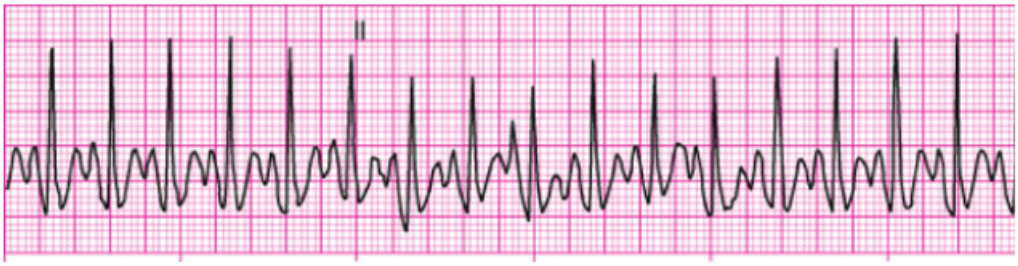


Figure 2.8. Sinus Tachycardia ([9])

Sinus Arrhythmia (SA): This arrhythmia is typically benign and does not require any sort of treatment. It is seen in children and also in mechanically ventilated patients.

Characteristic	Status
Rhythm	Regular
Rate	60-100/minute
p waves	Present, upright, symmetrical, one before every QRS
pri	.12-.20 seconds
QRS	.06-.10 seconds

Table 2.4. Characteristics of Sinus Arrhythmia

Figure 2.9. Sinus Arrhythmia ([9])

Wandering Atrial Pacemaker (WAP): It can be a normal aberration associated with Ischemia. There is no treatment required.

Characteristic	Status
Rhythm	Regular
Rate	60-100/minute
p waves	P waves vary in shape and size
pri	.12-.20 seconds
QRS	.06-.10 seconds

Table 2.5. Characteristics of Wandering Atrial Pacemaker

Figure 2.10. Wandering Atrial Pacemaker ([9])

2.5.2 Atrial

The rhythms that originate in the atrial will be examined in the following section. Premature atrial contractions, atrial flutter, atrial fibrillation, and supraventricular tachycardia are examples of these arrhythmias. The key characteristics of cardiac rhythms will be outlined, as well as nursing consequences and useful advice to help critical care nurses correctly interpret atrial arrhythmias.

Premature Atrial Contractions (PAC): it can be a normal aberration, Ischemia, or a signal of atrial irritability. It can lead to more serious atrial rhythms.

Characteristic	Status
Rhythm	Early beat (PAC) causes rhythm to be irregular
Rate	Underlying rhythm usually 60-100/minute
p waves	P waves have different configuration than underlying rhythm
pri	.12-.20 seconds in underlying rhythm
QRS	.06-.10 seconds in underlying rhythm

Table 2.6. Characteristics of Premature Atrial Contractions

Figure 2.11. Premature Atrial Contractions ([9])

Atrial Flutter (AFL): It is caused by electrolyte imbalance, Hypertension, Ischaemic heart disease, Congenital heart disease, and Rheumatic valve disease. Also after cardiac surgery.

Characteristic	Status
Rhythm	Regular or irregular
Rate	60-100/minute (ventricular rate) 250-400 (atrial rate)
p waves	No p waves present. Flutter waves (F waves) or 'sawtooth' waves
pri	No pri since no p waves
QRS	.06-.10 seconds

Table 2.7. Characteristics of Atrial Flutter

Figure 2.12. Atrial Flutter ([9])

Atrial Fibrillation (AF): It is caused by Electrolyte imbalance, Hypertension, Ischaemic heart disease, Congenital heart disease, and Rheumatic valve disease. Also following cardiac surgery.

Characteristic	Status
Rhythm	Irregular
Rate	60-100/minute (ventricular rate) >400/minute (atrial rate)
p waves	No p waves. Fibrillatory waves (f waves)
pri	No No pri since no p waves
QRS	.06-.10 seconds

Table 2.8. Characteristics of Atrial Fibrillation

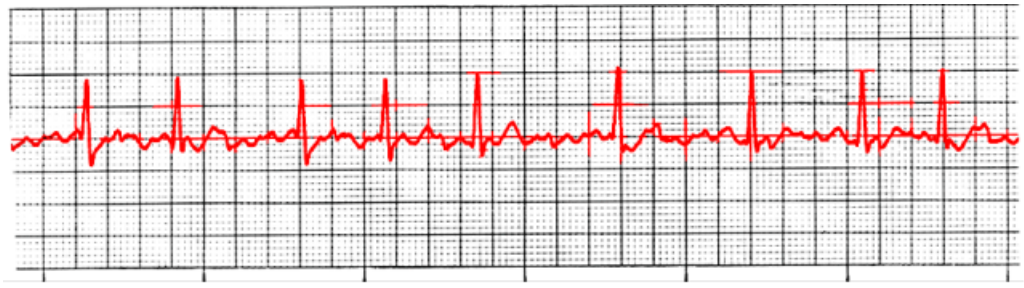


Figure 2.13. Atrial Fibrillation ([9])

Supraventricular Tachycardia (SVT): It is caused by Congenital, heart disease, Emotional stress, Physical stress or exertion, Illegal drugs (i.e. Cocaine or ecstasy), Alcohol, Caffeine.

Characteristic	Status
Rhythm	Regular
Rate	150-250/minute (atrial rate)
p waves	P waves may not be seen at higher rates
pri	.12-.20 seconds (if seen)
QRS	.06-.10 seconds

Table 2.9. Characteristics of Supraventricular Tachycardia

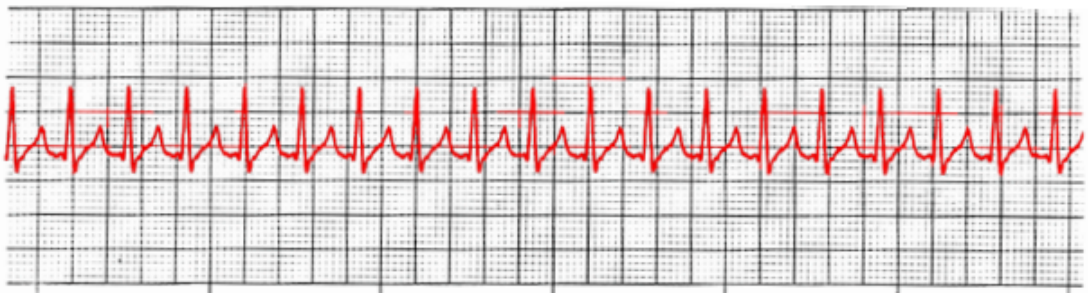


Figure 2.14. Supraventricular Tachycardia ([9])

2.5.3 Junctional

Junctional rhythms are temporary and non-lethal rhythms that originate in the AV node or junctional region. Inverted p waves are a typical feature of all junctional rhythms. Premature junctional contractions, junctional rhythm, and paroxysmal junctional tachycardia are among the rhythms covered in this section ([9]).

Premature Junctional Contraction (JPC): It is caused by Medication toxicity (i.e. digoxin), Ischemia. There is no treatment required. Continue to observe for an increasing number of JPCs since this indicates increasing AV node irritability.

Characteristic	Status
Rhythm	Early beat (PJC) causes the rhythm to be irregular
Rate	60-100/minute (underlying rhythm)
p waves	P waves inverted or not seen in JPC
pri	Not applicable
QRS	.06-.10 seconds (in underlying rhythm)

Table 2.10. Characteristics of Premature Junctional Contraction

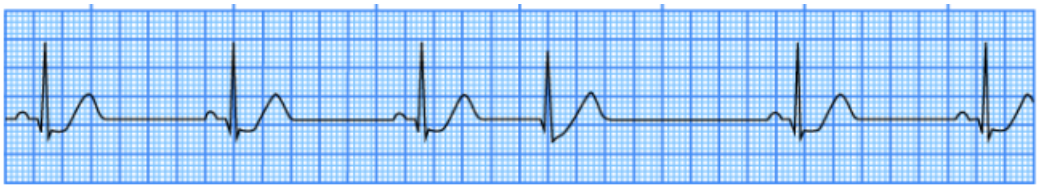


Figure 2.15. Premature Junctional Contraction ([9])

Junctional Rhythm (AVJR): It is caused by Medication toxicity (i.e. digoxin) or ischemia. It is necessary to treat causes.

Characteristic	Status
Rhythm	Regular
Rate	<60/minute
p waves	P waves inverted or absent
pri	.12-.20 seconds
QRS	.06-.10 seconds

Table 2.11. Characteristics of Junctional Rhythm

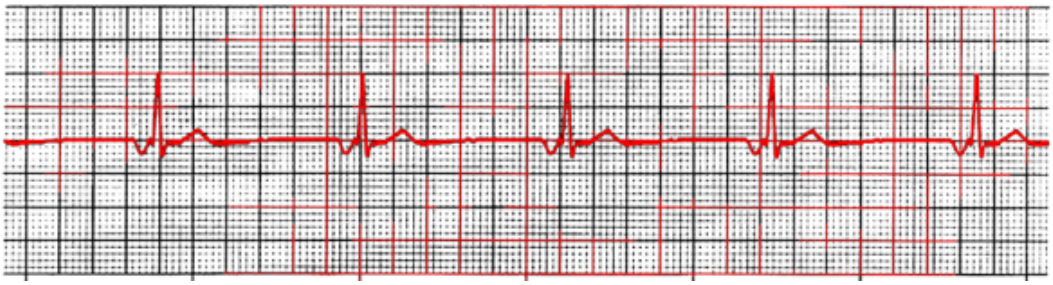


Figure 2.16. Junctional Rhythm ([9])

Accelerated Junctional Rhythm (AJR): It is caused by Medication toxicity (i.e. digoxin) or ischemia. It is necessary to treat causes.

Characteristic	Status
Rhythm	Regular
Rate	60-100/minute
p waves	P waves inverted or absent
pri	Not applicable
QRS	.06-.10 seconds

Table 2.12. Characteristics of Accelerated Junctional Rhythm

Figure 2.17. Accelerated Junctional Rhythm ([9])

Paroxysmal Junctional Tachycardia (JT): It is caused by ischemia. Its treatment is the same as SVT.

Characteristic	Status
Rhythm	Regular
Rate	150-250/minute
p waves	P waves inverted or absent (if seen)
pri	Not applicable
QRS	.06-.10 seconds

Table 2.13. Characteristics of Paroxysmal Junctional Tachycardia

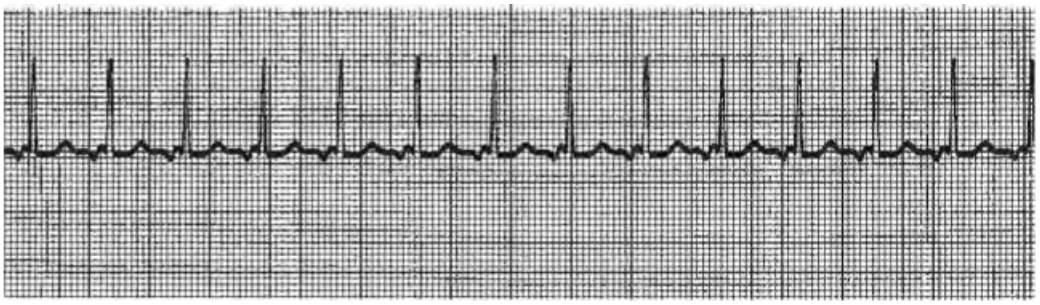


Figure 2.18. Paroxysmal Junctional Tachycardia ([9])

2.5.4 Ventricular

Premature ventricular contractions, ventricular tachycardia, and ventricular fibrillation are examples of ventricular rhythms that can be induced by irritability, as well as those that result from the failure of higher-level pacemakers. Irritability patients have significantly various treatment options and consequences.

Premature Ventricular Contractions (PVC): It is caused by Ventricular irritability (i.e.hypoxemia, acid-base imbalance, medications, electrolyte imbalance).

Characteristic	Status
Rhythm	Early beat (PVC) causes the rhythm to be irregular
Rate	60-100/minute (underlying rhythm)
p waves	None (in PVC)
pri	None (in PVC)
QRS	> .12 seconds (wide and bizzare)

Table 2.14. Characteristics of Premature Ventricular Contractions

Figure 2.19. Premature Ventricular Contractions ([9])

Ventricular Tachycardia (VTach): It is caused by Ventricular irritability (i.e.hypoxemia, acid-base imbalance, medications, electrolyte imbalance).

Characteristic	Status
Rhythm	Regular
Rate	150-250/min
p waves	None
pri	None
QRS	> .12 seconds (wide and bizzare)

Table 2.15. Characteristics of Premature Ventricular Contractions

Figure 2.20. Premature Ventricular Contractions ([9])

Ventricular Fibrillation (VF): It is caused by Ventricular irritability (i.e.hypoxemia, acid-base imbalance, medications, electrolyte imbalance).

Characteristic	Status
Rhythm	Irregular and chaotic
Rate	Cannot calculate
p waves	None
pri	None
QRS	None

Table 2.16. Characteristics of Ventricular Fibrillation

Figure 2.21. Ventricular Fibrillation ([9])

Idioventricular Rhythm (IR): It is caused by Ischemia, reperfusion post thrombolytics.

Characteristic	Status
Rhythm	Regular
Rate	<40/minute
p waves	No p waves
pri	No pri
QRS	> .12 seconds (wide and bizarre)

Table 2.17. Characteristics of Idioventricular Rhythm

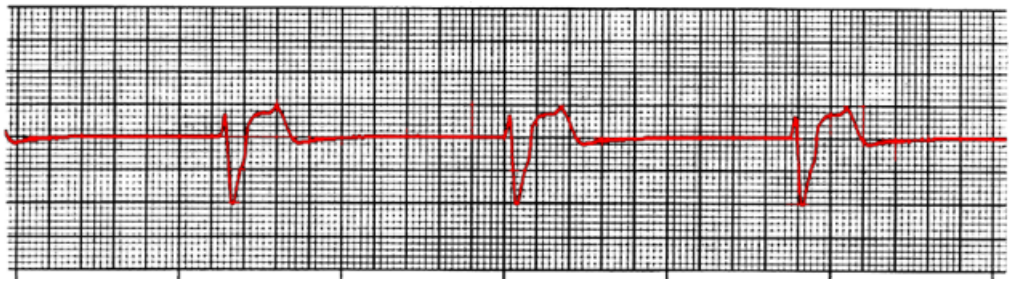


Figure 2.22. Idioventricular Rhythm

2.5.5 AV Blocks

Electrical conduction failure via the myocardium is characterized by atrioventricular (AV) blockages. Because AV blockages are linked to severe risk worsening or haemodynamic impairment, the critical care nurse must recognize and treat them as soon as possible. 1st-degree heart block, 2nd-degree heart block (Mobitz type 1 or Wenkebach), and 3rd-degree heart block are all types of AV block (complete heart block). ([9])

First Degree AV Block (IAVB): It is caused by AV nodal disease, Enhanced vagal tone (i.e. athletes), Myocarditis, Following Myocardial Infarction, Electrolyte disturbances, Medications (i.e. Calcium channel blockers, Beta blockers).

Characteristic	Status
Rhythm	Regular
Rate	60-100/minute
p waves	P waves normal
pri	>.20 seconds
QRS	.06-.10 seconds

Table 2.18. Characteristics of First Degree AV Block

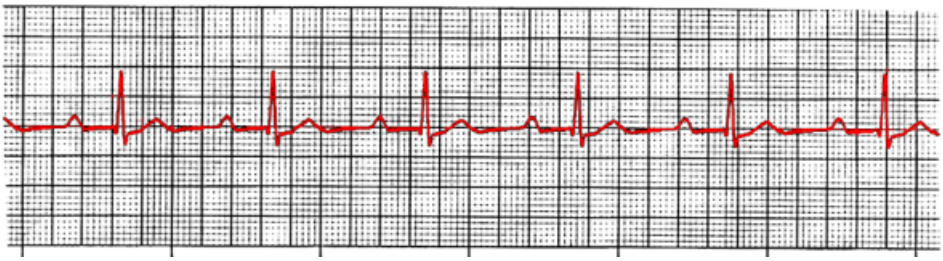


Figure 2.23. First Degree AV Block ([9])

Second Degree Type I (IIAVB): It is caused by Ischemia. Usually benign,

with no treatment required. If a patient becomes haemodynamically compromised interventions for bradycardia should be considered.

Characteristic	Status
Rhythm	Regular or slightly irregular
Rate	60-100/minute
p waves	P waves normal
pri	Progressively gets longer until a beat is dropped
QRS	.06-.10 seconds

Table 2.19. Characteristics of Second Degree Type I

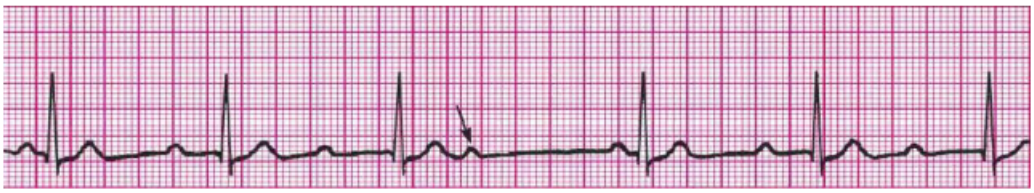


Figure 2.24. Second Degree Type I ([9])

Chapter 3

Related work and data employed

3.1 Related work

3.1.1 PhysioNet/Computing in Cardiology Challenge 2020

Introduction and keypoints

As I am going to use [11] PhysioNet/Computing in Cardiology Challenge 2020 dataset. It is often good to consider and find out what papers have been published and what kind of architecture they were following.

[12]To encourage more multidisciplinary researches, PhysioNet in Cardiology Challenge 2020 (Challenge 2020) [11] provided high-quality 12-lead ECG data obtained from multiple centers with a large set of cardiac abnormalities. The aim of Challenge 2020 was to identify clinical diagnoses from 12-lead ECG recordings, providing an opportunity to employ various advanced methods to address clinically important questions that are either unsolved or not well-solved [13].

Later I did the specific analysis specifically considering the papers published during the conference. 41 teams got selected during this challenge and others are simply discarded because the method did not work on the hidden set, the team failed to submit a preprint or a final article on time, or the team was absent during the CinC conference. Looking at the 41 teams' papers published in PhysioNet [11]. One can observe that some techniques are used by the majority of the teams. The results indicate that ECG classification is a complex process that includes multiple techniques. Among these techniques, signal processing, DNNs, convolutional neural networks (CNNs), and end-to-end and multi-binary classifications are used by all of the top 10 teams. In addition, there are several important points 1) deep-learning methods were more popular than traditional methods in Challenge 2020; 2) all the teams that employed deep-learning methods used CNNs; and 3) none of the top-10 teams used hand-labelled features (except demographic features); they all adopted

end-to-end models instead.

To investigate the techniques applied by each team, I considered five aspects of the methods that formed a solution pipeline (see 3.1): data preprocessing, feature engineering, machine-learning models, training strategy, and applications to the real world. Table 2 presents these five aspects.

Aspect	Inclusion	Usage(%)	# in top-10 methods	*p*-value
Data preprocessing	Signal processing	95.12	10	N.A.
	Data augmentation	31.70	6	0.071
	Imbalance handling	53.66	7	0.252
Feature engineering	Hand features	36.59	0	0.983
	Demographic features	29.27	5	0.109
Machine-learning models	Deep neural network	82.93	10	0.116
	Convolutional neural network	82.93	10	0.116
	Recurrent neural network/transformer	31.71	4	0.317
	Attention	24.39	6	0.006
Training strategy	Model ensemble	36.59	4	0.878
	End-to-End	80.49	10	0.139
	Multi-binary classification	58.54	10	0.002
Applications to the real world	Post-processing	2.38	1	N.A.
	Interpretability	4.76	0	N.A.
	Unknown classes and unseen patients	0	0	N.A.

N.A. means that the hypothesis test is not conducted.

Figure 3.1. Details of employed techniques.

[12] One can also notice that the three highest-ranking teams used the model ensemble [14, 15, 16], but only 14 out of 41 teams employed this strategy. It is also important to note that model ensemble only helps if used for a single model rather than models that are structurally different. Most of the team also used only age and sex as features rather other using demographic features or 12-lead ECG-based features. The training data in Challenge 2020 suffer from heavy class imbalance (as shown in 5.3) so most teams used threshold optimization [17, 15, 18] and weighted loss [19, 20] to handle imbalance class issue. In addition, over-sampling [21], down-sampling [22], and other methods have been employed in Challenge 2020.

Practical Lessons:

1. Data augmentation should be employed and adapted to specific scenarios.
2. Combining different features can improve performance.
3. A hybrid design of different types of deep neural networks (DNNs) is better than using a single type.
4. The use of end-to-end architectures should depend on the task being solved.
5. Multiple models are better than one.

Finding open source algorithms from published papers

To find the open source code of the papers I searched them on Google and Github, I was able to find the source code of 9 papers that were accepted during the challenge including team 2, team 6 and team 20. I will be later using these source codes for analysis to try to incorporate some additional features that I have identified during the centralized learning approach of Federated learning.

It is important to note that **Pytorch** is the most widely used framework by participants implementation of the Physionet Challenge 2020. [3.2]. 'Not Mentioned' either represents that they didn't write in their paper what kind of implementation they are following or I am not able to find out their implementation of the paper on the internet.

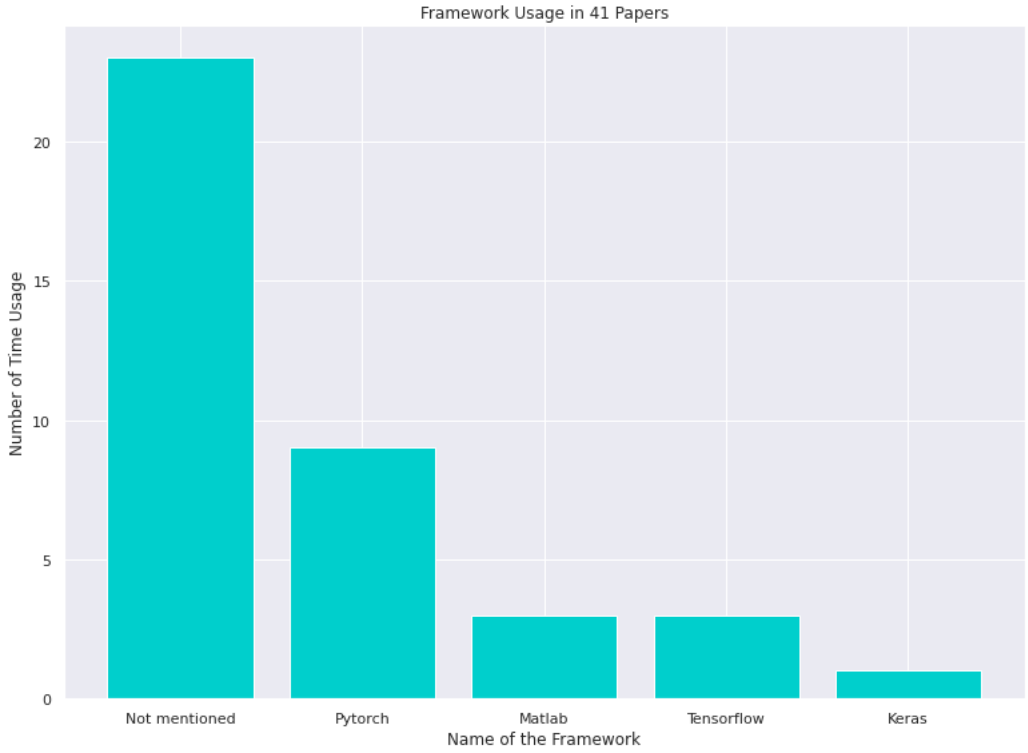


Figure 3.2. Most used framework during Physionet Challenge 2020

I have also created an excel sheet that has the combination of the meta-analysis of the 2020 challenge [12] and all the open source code related to each team that participated in the challenge. Which can be found here: https://bit.ly/3bgFtWs

3.2 Data employed

PhysioNet presents an annual series of biomedical 'Challenges' that focus on unsolved clinical and basic science challenges in collaboration with the annual Computing in Cardiology (CinC) conferences. The National Institutes of Health (NIH), Google, MathWorks, and the Gordon and Betty Moore Foundation have all lent their support to these challenges. George Moody, of the Laboratory for Computational Physiology (LCP), directed these Challenges for the first 15 years (from 2000 to 2014), before retiring due to ill health. Gari Clifford of Emory University and the Georgia Institute of Technology have been leading the Challenges since 2015. In 2021, the 'PhysioNet/Computing in Cardiology Challenges' was renamed the 'George B. Moody PhysioNet Challenges' to honour George's lifetime contributions to the discipline, particularly his seminal work on the Challenges [23].

The 2020 Challenge's purpose is to use 12-lead ECG records to detect clinically diagnosis. Starting from the clinical data provided, the participants must implement an open-source algorithm that can automatically classify the cardiac abnormality or abnormalities present in each 12-lead ECG recording and provide a probability or confidence score for each of them, with an emphasis on 27 diagnoses 3.1 To determine the winner, the trained models of the participants are run on hidden validation and test sets and their performance is evaluated using a novel, expert-based evaluation metric designed specifically for the 2020 Challenge. The team whose the algorithm that achieves the highest score is the winner of the Challenge.

Diagnosis	Code	Abbreviation
1st degree AV block	270492004	IAVB
Atrial fibrillation	164889003	AF
Atrial flutter	164890007	AFL
Bradycardia	426627000	Brady
Complete right bundle branch block	713427006	CRBBB
Incomplete right bundle branch block	713426002	IRBBB
Left anterior fascicular block	445118002	LAnFB
Left axis deviation	39732003	LAD
Left bundle branch block	164909002	LBBB
Low QRS voltages	251146004	LQRSV
Nonspecific intraventricular conduction disorder	698252002	NSIVCB
Pacing rhythm	10370003	PR
Premature atrial contraction	284470004	PAC
Premature ventricular contractions	427172004	PVC
Prolonged PR interval	164947007	LPR
Prolonged QT interval	111975006	LQT
Q wave abnormal	164917005	QAb
Right axis deviation	47665007	RAD
Right bundle branch block	59118001	RBBB
Sinus arrhythmia	427393009	SA
Sinus bradycardia	426177001	SB
Sinus rhythm	426783006	NSR
Sinus tachycardia	427084000	STach
Supraventricular premature beats	63593006	SVPB
T wave abnormal	164934002	Tab
T wave inversion	59931005	TInv
Ventricular premature beats	17338001	VPB

Table 3.1. Diagnoses, SNOMED CT codes and abbreviations for the 27 diagnoses that were scored for the Challenge.

The data are from five different sources:

1. CPSC Database and CPSC-Extra Database
2. INCART Database
3. PTB and PTB-XL Database
4. The Georgia 12-lead ECG Challenge (G12EC) Database
5. Undisclosed Database

The first source consists of three databases from the China Physiological Signal Challenge 2018 (CPSC2018), which took place in Nanjing, China at the 7th Interna-

tional Conference on Biomedical Engineering and Biotechnology [24]: the original public training dataset (CPSC), an unused dataset (CPSC-Extra), and the test dataset (the hidden CPSC set). The first two were shared as training sets, while the last one was split into validation and test sets for the 2020 Challenge. This training set consists of two sets of 6,877 (male: 3,699; female: 3,178) and 3,453 (male: 1,843; female: 1,610) of 12-15 ECG recordings lasting from 6 seconds to 60 seconds. Each recording was sampled at 500 Hz.

The second source is the public dataset from the St. Petersburg Institute of Cardiological Technics (INCART) 12-lead Arrhythmia Database [15 V. Tihonenko, A. Khaustov, S. Ivanov, A. Rivin, and E. Yakushenko, "St Petersburg INCART 12-lead arrhythmia database", PhysioBank, PhysioToolkit, and PhysioNet, 2008, DOI: 10.13026/C2V88N.]. The dataset was shared as a training set. This database consists of 74 annotated recordings extracted from 32 Holter records. Each record is 30 minutes long and contains 12 standard leads, each sampled at 257 Hz.

The third source from the Physikalisch Technische Bundesanstalt (PTB) includes two public databases which were shared as training sets: the PTB Diagnostic ECG Database and the PTB-XL, a large publicly available electrocardiography dataset. The first PTB database contains 516 records (male: 377, female: 139). Each recording was sampled at 1000 Hz. The PTB-XL contains 21,837 clinical 12-lead ECGs (male: 11,379 and female: 10,458) of 10-second length with a sampling frequency of 500 Hz.

The fourth source is the Georgia 12-lead ECG Challenge (G12EC) Database. This is a new database, representing a large population from the Southeastern United States, and is split between the training, validation, and test sets. The validation and test set comprised the hidden G12EC set. This training set contains 10,344 12-lead ECGs (male: 5,551, female: 4,793) of 10-second length with a sampling frequency of 500 Hz.

The fifth source is a dataset from an undisclosed American institution that is geographically distinct from the other dataset sources. This dataset has never been posted publicly and contains 10,000 ECGs all retained as test data [24]. As the mentioned dataset was not disclosed, I didn't use that one in my experiments.

The actual count of all the diagnoses by each database can be found in Figure 3.3. That was obtained by taking the first arrhythmia reported by each recording since each patient could contain more than one diagnosis based on its ECG.

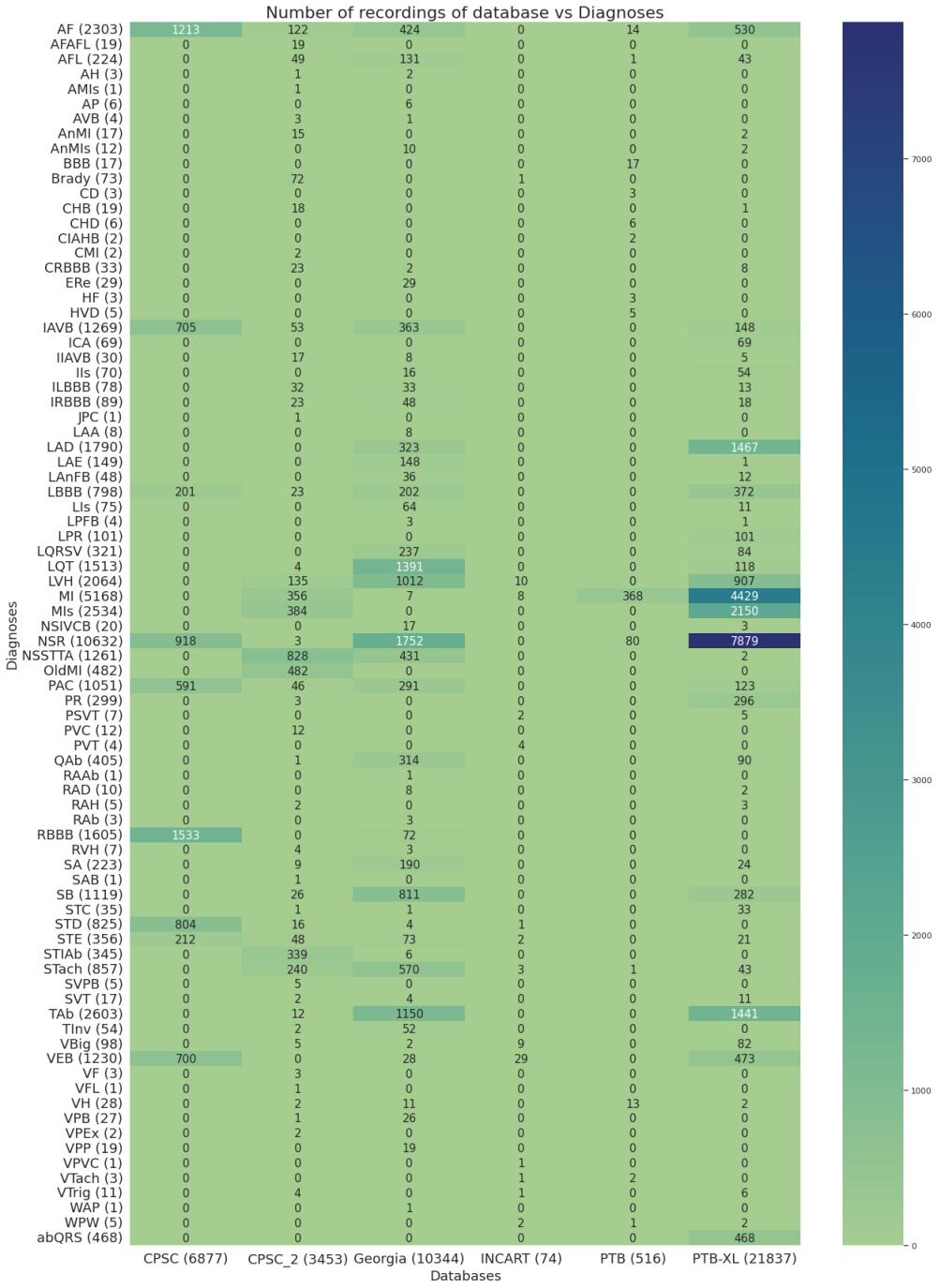


Figure 3.3. Number of recordings for each diagnosis by database

All data is provided in WFDB format. Each ECG recording has a binary **MATLAB v4 file** for the ECG signal data and a **text file** in WFDB header format describing the recording and patient attributes, including the diagnosis [25] [24] [26].

Chapter 4

Analytical techniques and tools

The parts that follow go over the most important analysis and approaches for studying, modelling, and predicting ECG arrhythmia diagnosis.

4.1 Commonly used techniques and tools

4.1.1 Data Wrangling (DW)

Data wrangling is the act of cleaning and combining chaotic and difficult data sets for easy access and analysis. With the amount of data and data sources growing all the time, it's more vital than ever to arrange massive volumes of data for analysis [27]. To facilitate data consumption and organization, this method normally requires manually transforming and mapping data from one raw format to another.

The most relevant Data Wrangling objectives are [27]:

- Collect data from a variety of sources to uncover "deeper intelligence."
- As soon as feasible, get reliable, actionable data into the hands of business analysts.
- Reduce the amount of time it takes to collect and organize jumbled data before it can be used.
- Allow data scientists and analysts to focus on data analysis instead of data manipulation.
- Encourage senior executives in a company to improve their decision-making skills.

Figure 4.1. Main steps in Data Wrangling

The data wrangling approach typically consists of six iterative steps, as seen in Figure 4.1, as mentioned by [28]:

1. **Publishing:** Data wranglers prepare data for downstream usage - whether by a specific user or program - and identify any special actions or logic that were employed to do so.
2. **Discovering:** Before delving into the data, it's important to first have a better knowledge of what's there since this will influence how you examine the data.
3. **Validating:** These are recurrent programming sequences that verify data quality, consistency, and security. Validation can include things like ensuring that qualities that should be distributed regularly are distributed uniformly.
4. **Enrichment:** "What more types of data can be obtained from what already exists?" one can question during the data wrangling stage. or "What further information could assist me in making better selections based on the current data?"
5. **Structuring:** The data must be structured in this step of data wrangling because raw data arrives in a range of formats and sizes.
6. **Cleaning:** By altering null values and establishing standard formats, data wrangling aims to improve data quality.

4.1.2 Feature Engineering (FE)

The act of choosing, altering, and transforming raw data into features that may be utilized in supervised learning is known as feature engineering. It may be

necessary to build and train better features for machine learning to perform well on new datasets.

Challenge features

Within the Physionet 2020, the organizers provided a code that calculated 14 features leveraged on the recordings. Those variables were based on the R-Peaks and the RR interval.

R-Peaks: It refers to the R wave's highest amplitude (as seen in Figure 2.4).

RR-Interval: On an ECG, it is the period between two consecutive R-waves of the QRS signal. The former is determined by the sinus node's inherent features as well as autonomic factors.

Then, with the previous measures, the competence calculated the mean, median, standard deviation, variance, skewness and kurtosis **ONLY** for the first lead. In addition, they used the age and sex provided with the initial raw data.

Spectral features

Leveraged on the solution developed by [29], I implemented 636 features that deal with the spectral part of the signals provided in the ECG. Spectral analysis (where the spectral features were derived) is a frequently utilized tool for exploring biomedical data. The waveform component forms, their time positions within the cardiac cycle, and the regularity of the heart period all influence the ECG signal's spectrum ([30]).

Usually, the Fourier Transform (FT) is used to extract information from signals like ECG. Nevertheless, the Fourier Transform has the drawback of capturing global frequency information or frequencies that are present throughout a whole signal. This type of signal decomposition may not be appropriate for many applications, such as electrocardiography (ECG), which involves signals with short periods of distinctive oscillation. The Wavelet Transform, which decomposes a function into a set of wavelets, is another option that corrects the FT approach [31].

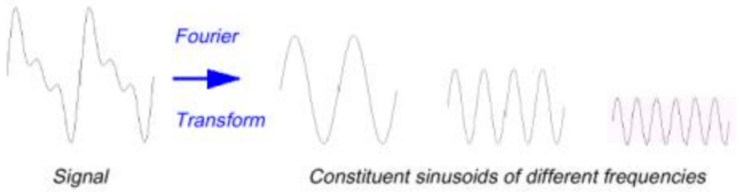


Figure 4.2. Wavelet representation

A Wavelet is a time-localized wave-like oscillation; an example is shown in Figure 4.2. Scale and location are the two most basic features of wavelets. The scale (or dilation) of a wavelet determines how "stretched" or "squished" it is. This attribute has to do with how waves are characterized in terms of frequency. The wavelet's position in time is defined by its location (or space).

Then, the schema of features calculated is as follows. For each lead calculate:

1. **Statistics**: Percentiles (5, 25, 50, 75, 95), mean, standard deviation and variance for the complete signals.
2. Calculate **coefficients** of **Discrete Wavelet Transform (DWT)**. DWT gets local frequencies for the signals. The Coefficients are calculated using the function `wavedec` from the Python's library `pywt`.
3. For each **coefficient** of DWT calculate:
   - **Statistics**: Percentiles (5, 25, 50, 75, 95), mean, standard deviation and variance.
   - **Shannon's entropy (same that entropy)**: It's related to the "amount of information" of a variable. In other words, it measures the information of the distribution.

Spectral features morphology

$$feature_s = l(Lead_i)\_c\_(Coefficient_j)\_(operation_k) \tag{4.1}$$

Each 636 spectral feature is based on *lead* as l, their *coefficient* as c and *operation* applied on it for example: mean, coefficient percentiles, standard deviation etc. Also, $lead_i$ represents ECG lead number from 00-11, $Coefficient_j$ represents a coefficient number from Discrete Wavelet Transform(there are 5 coefficients in total 1-5) and $operation_k$ represents the operation name like mean as average, standard deviation as std, variance as var, percentiles represents as n5(percentile 5), n25(percentile 25), n50(percentile 50), n75(percentile 75), n95(percentile 95) and entropy applied on coefficient get represented as c1-c5. At the end from each ECG lead, we get 53 features and in 53x12 we get 636 spectral features in total. Below in figure [5.6] names of features are represented by above represented [4.1] morphology.

4.1.3 Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to identify patterns, spot anomalies, test hypotheses, and check

assumptions using summary statistics and graphical representations. It's important to initially comprehend the data before attempting to get as many insights as possible. EDA is all about making sense of data before getting their hands dirty with it. The major steps commonly examined in an EDA are shown in Figure 4.3.

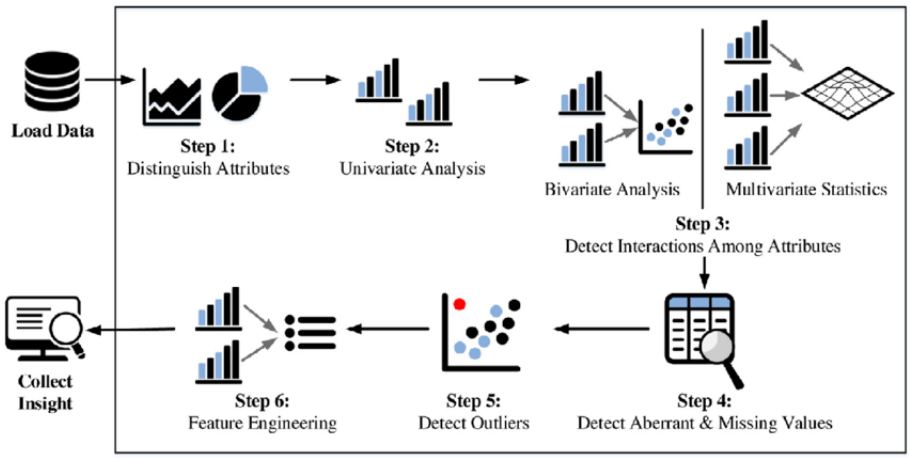


Figure 4.3. Schema of a EDA

4.1.4 Unbalanced classes

One of the most difficult issues when training a model is modelling imbalanced data [32]. When dealing with classification problems, the intended class balance is quite important. When a dataset has an uneven distribution of classes, the models attempt to learn only the dominant class, resulting in biased predictions.

One approach for addressing this issue is random sampling. Random resampling can be accomplished in two ways, each with its own set of benefits and drawbacks:

- **Oversampling:** Replicating examples from the minority class.
- **Undersampling:** Deleting examples from the majority class.

To put it another way, both oversampling and undersampling include creating bias by selecting more instances from one class than from another. The prior is used to compensate for an imbalance that is already present in the data or that is likely to occur if a perfectly random sample is obtained [33]. Because it makes no assumptions about the data, random sampling is a naive strategy. To minimize the data's influence on the Machine Learning algorithm, a fresh adjusted version of the data with a new class distribution is generated.

Random Oversampling and SMOTE were the two oversampling techniques chosen for this project. The synthetic Minority Oversampling Technique is a technique for creating synthetic samples for the minority class. Overcoming the problem of

overfitting produced by random oversampling is easier with this method. It focuses on the feature space to generate new examples by interpolating between positive occurrences that are near in proximity.

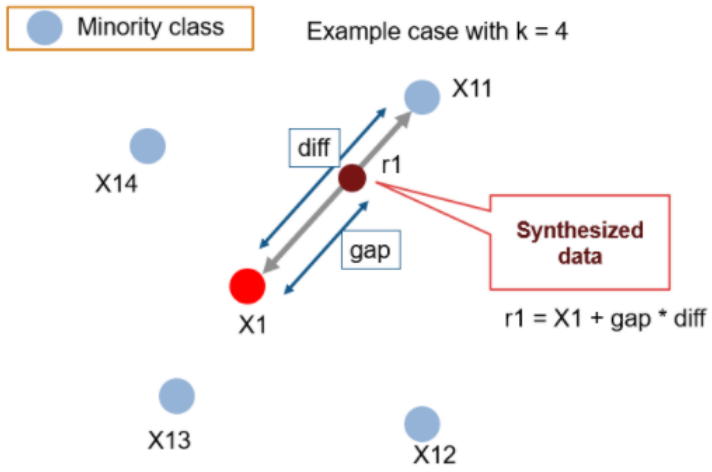


Figure 4.4. SMOTE process illustration

SMOTE uses the k-nearest neighbour technique to create synthetic data. To make them follow the instructions below. [32]:

1. Find the nearest neighbours of the feature vector.
2. Determine the distance between the two sample sites.
3. At random, the distance is multiplied by an integer between 0 and 1.
4. Find a new point on the line segment at the calculated distance.
5. Rep the procedure for each of the feature vectors that were discovered.

4.1.5 Machine Learning Models

Classifiers are the models provided in the following sections. These tools were created to determine which behaviours are more likely to be associated with various arrhythmia patterns. Each of these methods is widely utilized in various data-driven systems, and they have demonstrated useful behaviour in a variety of classifying tasks, including ECG classification.

The various versions of the dataset were created using Python Notebooks in Google Colab. This section will detail the key models that were tested and evaluated.

Model 1 - (XGB) XG-Boost algorithm

The XG-Boost technique, which has proven to be effective in a variety of classification and regression problems, is the first attempt to classify ECG signals. The aforementioned algorithm has been used in a variety of sectors, including economics, credit rating, and health-related difficulties. The preceding are reasons to expect that such a strategy will be effective in the field of arrhythmia detection today.

XG-Boost is a decision-tree-based ensemble Machine Learning approach that uses gradient boosting ([34] [35]). When it comes to unstructured data prediction, *Artificial Neural Networks* outperforms all other algorithms or frameworks (text, audio, pictures, etc.). However, for small-to-medium tabular data, such as the one utilized in this challenge, *decision tree-based* algorithms are now rated best-in-class.

XG-Boost minimizes a loss function to provide an additive expansion of the objective function, similar to gradient boosting. Because XG-Boost is only interested in decision trees as base classifiers, the complexity of the trees is controlled using a variation of the loss function.

$$L = \sum_{i=1}^{n} L(y_i, \hat{y_i}) + \sum_{k=1}^{K} \Theta(p_k) \tag{4.2}$$

$$\Theta(w) = \gamma Z + \frac{1}{2}\lambda ||w||^2 \tag{4.3}$$

The number of leaves on the tree is Z, and the leaf output scores are w ([35]). This loss function can be included in the split criterion of decision trees, resulting in a pre-pruning strategy. Trees with a greater γ value are easier to understand. The amount of loss reduction gain required to separate an internal node is determined by γ ([34]). Shrinkage is a regularization parameter in XG-Boost that decreases the step size in the additive expansion. Finally, other techniques such as tree depth can be utilized to keep the trees from becoming too complex. As a result of lowering tree complexity, the models are trained faster and need less storage space.

Model 2 - (Catboost) Catboost

The second candidate for predicting the arrhythmia type for ECG is the Catboost algorithm. The latter is a decision tree gradient boosting technique. It was created by Yandex (with its final version in 2017) researchers and engineers and is used by Yandex and other firms such as CERN, Cloudflare, and Careem taxi for search, recommendation systems, personal assistant, self-driving cars, weather prediction, and many other activities. Anyone can use it because it is open-source.

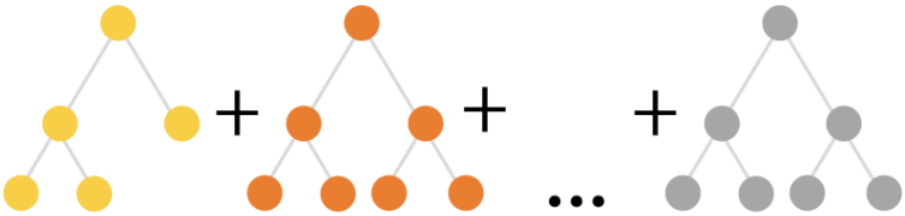


Figure 4.5. Catboost (decision trees) illustration

The implementation of ordered boosting [36], a permutation-driven alternative to the conventional approach, and a novel technique for processing category characteristics are two key algorithmic innovations offered in CatBoost. Both strategies were developed to combat a prediction shift induced by a specific type of target leakage found in all current gradient boosting algorithm implementations.

Model 3 - (DNN) Deep Neural Networks

A Deep Neural Network is another method for predicting ECG diagnosis. A DNN is a set of algorithms that attempts to recognize relationships in a batch of data by mimicking how the human brain functions.

In this context, deep neural networks refer to organic or artificial systems of neurons ([37]). Deep neural networks can adapt to changing input and produce the best possible result without requiring the output criteria to be modified because they can adapt to changing input. Neural networks, an artificial intelligence-based concept, are swiftly gaining popularity in the development of trading systems.

Neural networks aid in time-series forecasting, algorithmic trading, securities classification, credit risk modelling, and the generation of proprietary indicators and price derivatives in the financial world ([38] [39]). The deep neural network of the human brain is akin to a neural network. A "neuron" in a deep neural network is a mathematical function that collects and categorizes data according to a set of rules. The network closely resembles curve fitting and regression analysis, two statistical methods.

Perceptrons are grouped in interconnected layers in a multi-layered perceptron (MLP) [39], as indicated in Figure 4.6 . The input layer is responsible for collecting input patterns. In the output layer, input patterns can be mapped to classifications or output signals. Hidden layers fine-tune the input weightings until the neural network's margin of error is as little as possible. Hidden layers are supposed to deduce salient elements from input data that can predict outcomes. This is how feature extraction works, and it's similar to how statistical methods such as principal component analysis function ([39]).

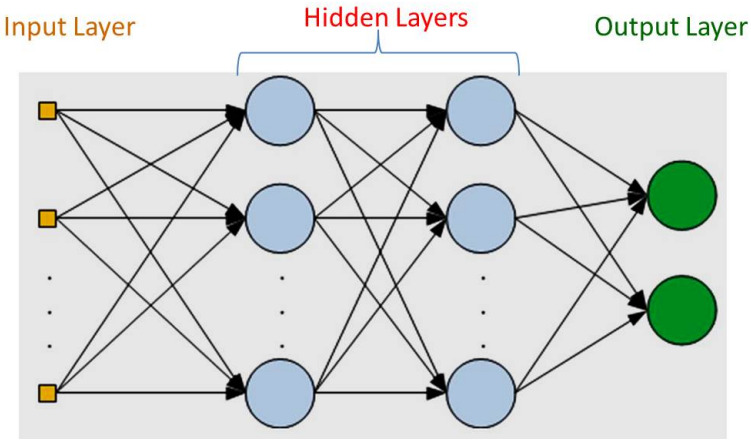


Figure 4.6. Deep Neural Network (Multi-layer Perceptron) schema

Model 4 - (LSTM) Long-Short Term Memory

Long short-term memory networks are a type of Deep Learning network. It's a class of recurrent neural networks (RNNs) that can learn long-term dependencies, which is useful for solving sequence prediction issues. Apart from single data points like photos, LSTM has feedback connections, which means it can process the complete sequence of data.

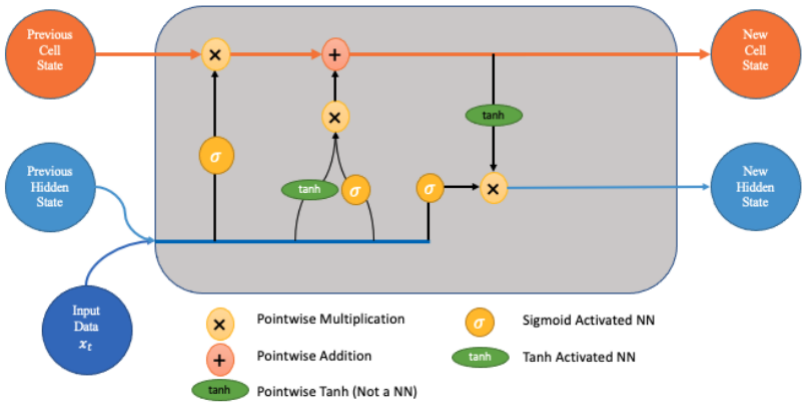


Figure 4.7. LSTM general schema

An LSTM model's primary role is played by a memory cell called a 'cell state,' which maintains its state across time. The horizontal line that runs through the top of the diagram below represents the cell state. It can be compared to a conveyor belt on which data just passes, unmodified [40].

Model 5 - Challenge Team 2 - ResNet with an SE layer[15]

Challenge team 2 designed a modified ResNet with larger kernel sizes that model long-term dependencies. They embedded a Squeeze-And-Excitation layer into the modified ResNet to learn the importance of each lead adaptively.

After obtaining the input signals, they designed ResNet to assign the 12-lead ECG recordings to the 24 diagnostic classes.

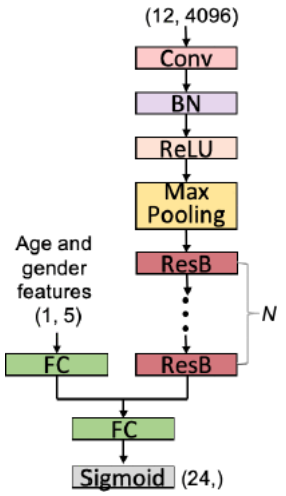


Figure 4.8. The proposed network architecture.

As shown in [4.8] the improved ResNet consists of one convolutional layer followed by N = 8 residual blocks (ResBs), each of which contains two convolutional layers and a squeeze and excitation (SE) block (Fig. 2). Since some samples have multiple classes of 27 clinical diagnoses, instead of using the softmax function in a traditional classification problem, they assumed that each class was independent and used the sigmoid function for each output neurons to cope with this multi-task problem.

The first (convolutional) layer and the initial two ResBs units have 64 convolution filters. The number of filters increases by a factor of two for every second ResB unit. The feature dimension is halved after the max pooling layer, and the third, fifth, and seventh ResBs.

The improved ResNet has four modifications from the original ResNet [41]. First, they modified the final fully connected (FC) layer to incorporate patient age and gender. These two features were passed through another FC layer with 10 neurons before inclusion in the final layer. Second, they used a relatively large kernel size of 15 in the first convolutional kernel, and a large kernel size equal to 7 in the latter convolutional kernels. Previous work has shown that large kernel sizes are more helpful for networks to learn meaningful features [42].

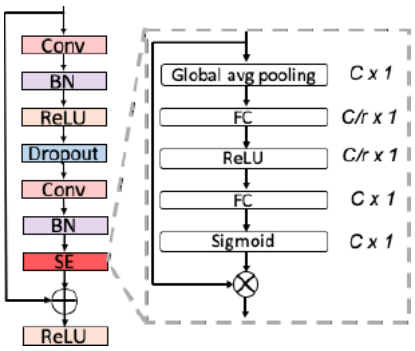


Figure 4.9. The proposed network architecture.

Third, as shown in [4.9], they added a dropout layer with a dropout rate of 0.2 between two convolutional kernels in each ResB to reduce the likelihood of overfitting. Finally, They added a SE block into each ResB depicted in [4.9]. The SE block has been to model channel inter-dependencies, and in this case, they incorporate it to model the spatial relationship between the ECG channels. The SE block, introduced by [43] uses a multi-layer perceptron (MLP) with one hidden layer to calculate the importance of the channels. The parameter r = 16 in Fig. 2 denotes the reduction factor, which controls the capacity of the MLP.

Model 6 - Challenge Team 20 - A hybrid model (CNN + Rule Based)[15]

Team 20 developed a hybrid model combining a rule-based algorithm with different deep learning architectures. They compared two different Convolutional Neural Networks, a Fully Convolutional Neural Network and an Encoder Network, a combination of both, and with the addition of another neural network using age and gender as input. Two of these combinations were finally combined with a rule-based model using derived ECG features.

CNN Architectures:

As a starting point for classifying the ECG signals, they employed FCN and Encoder types of CNN models as described in [44]. Two models were tested without any modifications to the architecture other than changing the input and output layers to fit our input data and output classes. All output layers of each model used a Sigmoid activation function.

To make use of the provided age and gender data, a simpler neural network model with 2 inputs, one hidden layer of 50 units, and 2 outputs in the final layer was added. This new model was combined with our FCN and Encoder models by concatenation of the last layer of the CNNs.

Age and gender data were passed into the simple neural network as integers, but in some information files, the age of the patient was not given and was assigned a

value of -1. The gender data was transformed into integers, where a male was set equal to 0, a female equal to 1, and an unknown was set to 2.

The two CNN models (FCN and Encoder) were combined as parallel models, concatenated on the second last layer. This model was tested with and without a parallel dense layer1.

Rule-Based Model

The rule-based algorithm used the raw ECG signal, without any padding or truncating, as input. R-peak detection [45], and heart rate variability (HRV) analysis was programmed to add relevant derived features to the algorithm. An HRV score was obtained by computing the root mean square of successive differences between normal heartbeats (RMSSD) using the detected R-peaks as timing indicators of each heartbeat.

The rule-based algorithm was able to classify eight different diagnoses: atrial fibrillation, bradycardia, low QRS-complex, normal sinus rhythm, pacing rhythm, sinus arrhythmia, sinus bradycardia, and sinus tachycardia.

The rule-based algorithm performed classification independent of the deep learning models. If there was disagreement between the rule-based algorithm and the CNN model, the rule-based algorithm overwrote the classification from the CNN model.

4.1.6 Metrics

It is vital to create metrics that will assist in determining whether a model is better than others to determine whether it is better than others. There are explanations for each of the metrics used in the following sections.

Confusion Matrix

A confusion matrix, like the one shown in table 4.1, demonstrates how well a classification model works on test data for which the true values are known ([46]). The confusion matrix is simple in itself, but the related nomenclature can be confusing. In the following examples, I've created a hypothetical target variable called "Diagnose A" with the values "Yes" (if the recording belongs to that diagnosis) and "No" (if the recording does not belong to that diagnosis).

	Actual Class	
Predicted Class	Diagnose A - YES = 1	Diagnose A - NO = 0
Diagnose A - YES = 1	True Positives (**TP**)	False Positives (**FP**)
Diagnose A - NO = 0	False Negatives (**FN**)	True Negatives (**TN**)

Table 4.1. Confusion Matrix representation

Here is an explanation for each of the matrix's elements to understand the preceding terminology ([46] [47]).

- *True negatives (TN):* The model predicted they wouldn't have diagnosed A, and they don't.
- *True positives (TP):* These are examples when the model predicted yes (the recording has the diagnose A), and they don't.
- *False positives (FP):* The model projected that they would have diagnosed A, but they don't. (This is also referred to as a "Type I error.")
- *False negatives (FN):* The model anticipated that they would not have diagnosed A, yet they do. (This is often referred to as a "Type II error.")

Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.4}$$

The most basic performance metric is accuracy, which is defined as the proportion of correctly predicted observations to all observations. If a model is correct, one would assume it is the best. Yes, accuracy is a relevant measure when the datasets are symmetric and the number of false positives and false negatives is about equal.

Imagine the case when the training set contains 98 percent samples of class A and 2% samples of class B, for example. The model may thus easily attain a 98% training accuracy by simply guessing every training sample that belongs to class A. When the same model is tested on a test set that contains 60% class A samples and 40% class B samples, the test accuracy reduces to 60%. As a result, classification accuracy is poor, but it gives the image of great accuracy.

Then, when the cost of misclassification of minor class samples becomes significant, ([47]) the true issue appears. The cost of failing to diagnose, for example, a sick person's ailment is significantly greater than the expense of submitting a healthy person to additional tests when dealing with a rare but lethal disorder.

Precision

$$Precision = \frac{TP}{TP + FP} \tag{4.5}$$

Precision [46] is the ratio of accurately predicted positive observations to total expected positive observations. This measure answers the question of how many

of the drivers who were identified as drowsy drove. Precision is linked to a low false-positive rate.

Precision is a good statistic to employ when the costs of False positives are high. Take, for example, the identification of email spam. In email spam detection, a false positive happens when an email that is not spam (actual negative) is wrongly identified as spam (predicted spam). If the precision of the spam detection model is low, the email user may miss important emails.

Recall

$$Recall = \frac{TP}{TP + FN} \tag{4.6}$$

Recall [46] is the ratio of successfully predicted positive observations to all observations in the actual class. It's meant to answer the question of how many drivers who slept were labelled as such.

In the case of identifying sick patients, for example, if a sick patient (Actual Positive) conducts the test and is predicted to be healthy (Predicted Negative). The cost of a False Negative will be quite high if the condition is infectious.

F1 Score

$$F1Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \tag{4.7}$$

The F1-Score is the weighted average of Precision and Recall. As a result, both false positives and false negatives are taken into account in this score. F1 is often more valuable than accuracy, despite the fact that it is less intuitive ([47] [46]). This is especially true if the class distribution is unequal. When the costs of false positives and false negatives are equal, accuracy works well. If the cost of false positives and false negatives differs significantly, it is best to evaluate both Precision and Recall.

Overall index

As a final metric, I introduced the use of the mean value of Accuracy, Precision, Recall and F1-Score. With the latter, it is possible to check with only one metric the overall behaviour of the classifiers.

4.2 Edge computing(EC) fundamentals

4.2.1 Definition

Edge computing is a distributed computing paradigm that brings computation and data storage closer to the sources of data. This is expected to improve response times and save bandwidth. It is an architecture rather than a specific technology. It is a topology- and location-sensitive form of distributed computing.

Edge computing is primarily concerned with transmitting data among the devices at the edge, closer to where user applications are located, rather than to a centralized server (see 4.10). The edge node (or edge client, edge device) is usually the resource-constraint device that the end user uses and it is geographically close to the nearest edge server, who has abundant computing resources and high bandwidth communicating with end nodes. When the edge server requires more computing power, it will connect to the cloud server. The most important consequences of this architecture are twofold: latency is dramatically reduced as data does not need to travel as far, and bandwidth availability improves significantly, as the user is no longer relying on sharing a single traffic lane to transfer their data. Indeed, this new computing paradigm offers great cost savings for companies that do not have the resources to build dedicated data centers for their operations. Instead, engineers can build a reliable network of smaller and cheaper edge devices. [3]

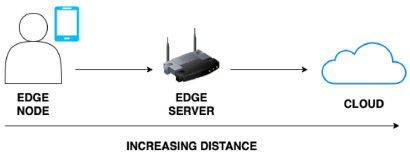


Figure 4.10. Edge Computing Overview

In addition, federated learning has been discussed a lot recently. It is a collaborative machine learning framework allowing devices from different resources with different private datasets working together to study and train a global model. Federated learning can not only collaborate the computational resources from different devices but also preserve privacy at the same time.

Given the common features of both edge computing and federated learning, edge computing is a naturally suitable environment to apply the federated learning framework. Therefore, edge federated learning is more and more appealing in both academic research and industry in recent days. Here, we first have a brief introduction to edge computing and federated learning respectively and discuss them in detail later.

4.2.2 Reasons for Edge Computing:

There are a set of key reasons why industry executives are transitioning from a traditional cloud-based model to edge computing platforms. The two major factors that were already discussed beforehand are low latency and high bandwidth [48]. However, the edge also provides for greater security. For example, sending data to an edge device will give any potential attackers less time to launch an attack as compared to the cloud simply because the latency is lower. Moreover, attacks like DDoS that would normally be debilitating in a cloud-based environment are rendered almost harmless in an edge computing environment because the affected edge devices can be removed from the network without hampering the overall functionality of the network as a whole. Of course, this also means that edge networks are much more reliable as they do not have a single point of failure. As discussed briefly beforehand, edge networks are much more easily scalable because the devices have much smaller footprints. Indeed, a scale-out strategy of scalability rather than a scale-up one offers companies a very attractive way of getting good performance at a low cost. Moreover, some of these edge devices or edge data centers may not even need to be built from scratch by any one company. Different stakeholders can partner up to share the resources from the already existing IoT devices in the edge network.

4.2.3 EC Operating Principles:

To deliver these benefits to end-users, engineers have relied on a common set of key operating principles when building edge computing systems [49]:

Mobility: For applications like self-driving cars, the edge devices have to accommodate a constantly moving end-user without sacrificing latency or bandwidth. Some approaches solve this problem by positioning edge devices on the roadside.

Proximity: To deliver low latency guarantees, the edge devices must be positioned as close as possible to the end users. This could mean performing computation directly at the edge device or investing in a local edge computing data center that is close to the end-user.

Coverage: For edge computing to become ubiquitous, network coverage must be far-reaching. Thus, the exact distribution of nodes in an edge computing framework is imperative to achieving an optimal user experience. Of course, a dense distribution is preferred, but this must be balanced with cost constraints.

4.3 Federated learning (FL) fundamentals

4.3.1 Definition

Federated learning (FL) is a machine learning technique for training machine learning models cooperatively on several devices or local servers in a decentralized way, preserving data privacy and data ownership for the device/server owner [50]. FL is extremely advantageous for highly decentralized healthcare data, especially with the growing prevalence of IoT devices for continuously capturing data and monitoring health.

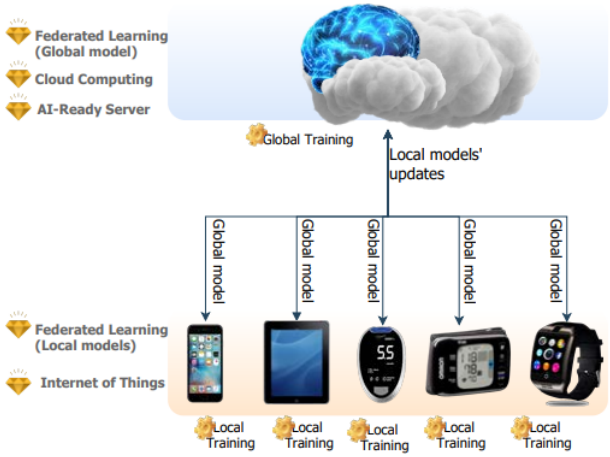


Figure 4.11. FL framework Overview

Figure 4.11 depicts a high-level view of the framework and how the technologies will interact together. The IoT devices will collect data from users and train a local deep learning model that is a copy of a global model that was previously received. Following the completion of the local training phase, the models will collaborate to train a global model utilizing their updates rather than the raw data provided by the users. These model updates indicate changes in the weights of the models during the training process and do not reflect any private or personal information about the users.

All participating models will send updates to a cloud server, where they will be compiled and used to train the global model [51] [52]. Each device will receive a new copy of the updated global model once the global model training procedure is completed. As a result, the models will be trained and updated regularly without sharing any personal information. As a result, the framework will support an IoT-based decentralized architecture in which models are spread among IoT devices without the need for a centralized server to operate the model and serve users. It will also protect users' privacy by processing and analyzing their data on IoT devices without disclosing it.

4.3.2 FL types

FL is divided into five categories [53] based on data partitioning, machine learning models (ML Models), privacy mechanisms, communication architecture, and federation scale.

Data Partitioning

The datasets of various clients share the same properties in **Horizontal data partitioning** [54], however, there is limited sample space intersection. All FL architectures use horizontal partitioning the most. Aggregation at the server is made easier by the fact that a standard model can be used for all clients; FedAvg is typically used for aggregation. A dataset containing ONLY breast cancer patients from a specific hospital would be simple to comprehend example.

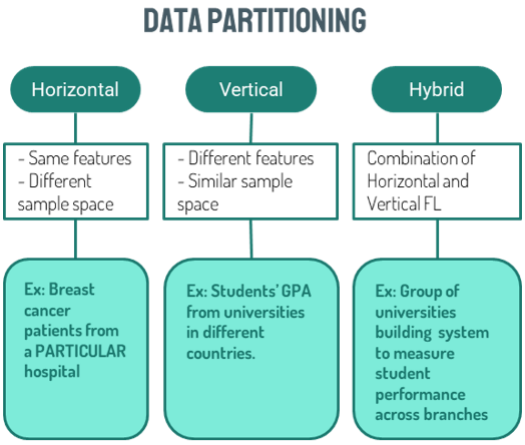


Figure 4.12. Data partition-based FL types

When clients are exposed to distinct feature spaces but the same or similar sample space, **Vertical Data Partitioning** comes into play. Entity alignment algorithms are utilized to find overlapping samples among the client data, and this overlapped data is used for training [53]. A dataset of students' GPAs obtained from institutions across the globe is a nice example. The feature space, which includes the grading scale and evaluation measure, is distinct.

Horizontal and vertical data partitioning are combined in **Hybrid Data Partitioning**. A set of universities intending to develop an FL System to assess student achievement across branches is an easy to comprehend case for hybrid partitioning.

ML models

The issue statement and dataset are frequently used to determine the machine learning models to use [53]. One of the most widely used models is neural networks (NN). Apart from NNs, decision trees are also used, as they are highly efficient and simple to understand. Models can be **homogeneous** or **heterogeneous** in a FL system.

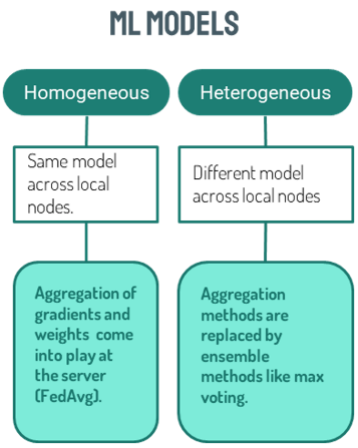


Figure 4.13. ML models-based FL types

In the case of the former, all clients use the same model, while the server uses gradient aggregation. In the latter instance, however, there is no possibility of aggregating because each client has a unique model. Aggregation methods are substituted with ensemble methods like max voting at the server in the case of heterogeneous models [53].

Privacy Mechanisms

The most controversial part of FL is how it deals with privacy. The main concept is to prevent client information from leaking out. The server may decipher the data of clients without encryption by applying learning gradients. As a result, it's critical to hide the gradients. Differential privacy and cryptographic approaches are commonly used to address privacy concerns in FL systems.

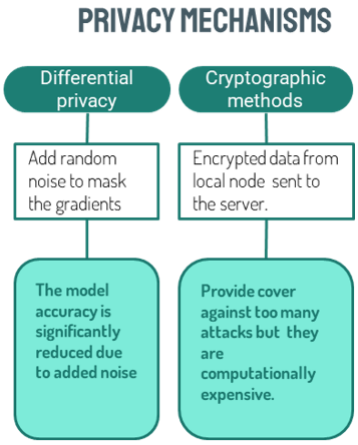


Figure 4.14. Privacy Mechanisms-based FL types

Differential privacy is a technique for hiding gradients by adding random noise to data or model parameters. Due to the extra noise, this strategy has a considerable negative in terms of model accuracy.

In FL systems, **cryptographic approaches** such as homomorphic encryption and safe multi-party computation is commonly used. The process is straightforward: clients send encrypted data to the server, the server processes the data, and then the encrypted output is decrypted to obtain the final result. Even though these methods protect against a wide range of threats, they are computationally intensive.

Architecture

There are two types of FL system architecture: centralized and decentralized. Both types of architecture work in the same way; the only difference is in client-server communication. We have a second model that acts as a server in a **centralized architecture**, and all parameter updates are done in this global model.

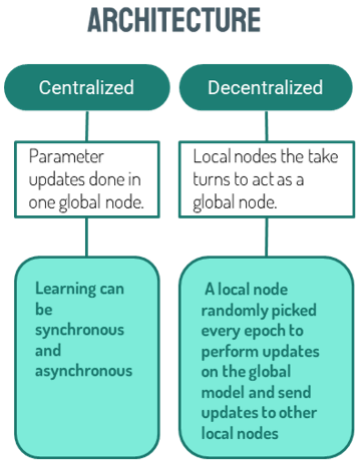


Figure 4.15. Architecture-based FL types

In a **decentralized design**, on the other hand, clients take turns acting as servers. Every epoch, a client is chosen at random to make global model changes and send the global model to other clients.

Scale of Federation

The scale of the federation can be divided into two types: cross-silo and cross-device. To grasp the distinction between the two, relate cross-silo with organizations and cross-devices with mobiles. When using **cross-silo**, the number of clients is usually minimal, but they have a lot of computing power.

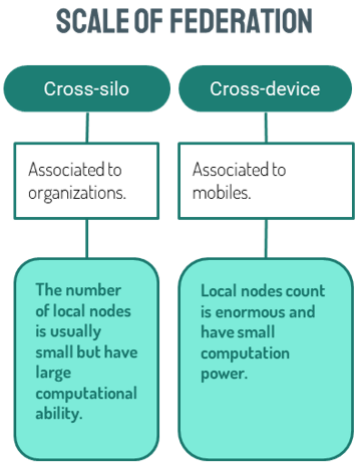


Figure 4.16. Scale of federation-based FL types

Regarding **cross-device**, the number of clients is enormous, but their computing power is limited. Another consideration is reliability: while we can rely on organiza-

tions (cross-silo) to be ready to train at all times, this is not the case with mobile phones (cross-devices). There's a chance that a bad network will make the gadget unavailable.

4.3.3 Advantages and disadvantages

FL has a lot of advantages over traditional, centralized systems [55]. Some of the most remarkable upper hands of FL are:

- **Data security**: Keeping the training dataset on the devices eliminates the need for a data pool for the model.
- **Data diversity**: Companies may be unable to merge datasets from diverse sources due to challenges other than data security, such as network unavailability in edge devices. Federated learning makes it easier to access diverse data, even when data sources can only interact at particular periods.
- **Continuous learning in real time**: Models are continuously enhanced utilizing client input, eliminating the requirement to aggregate data for continuous learning.
- **Technology efficiency**: Because federated learning models do not require a single complex central server to evaluate data, this technique requires less complex hardware.

On the other hand, FL needs to deal with some relevant challenges. The most common are:

- **Investment requirements**: FL models may necessitate frequent communication between nodes, which may necessitate an investment. This means that high bandwidth and storage capacity are among the system requirements.
- **Data Privacy**: In FL, data is not collected on a single entity/server; instead, numerous devices are used to collect and analyze data. Even though only models, not raw data, are transferred to the central server, models can be reverse engineered to identify client data, thereby increasing the attack surface. Differential privacy, secure multiparty computation, and homomorphic encryption are examples of privacy-enhancing technologies that can be utilized to improve the data privacy capabilities of federated learning.
- **Performance limitations**: In FL, models from several devices are combined to create a superior model. Device-specific factors may hinder the generalization of models from some devices, lowering the accuracy of the model's next

generation. Researchers investigated scenarios in which one of the federation's members could use secret backdoors in the joint global model to intentionally attack others.

4.3.4 Proposed approach

After understanding the theory and characteristics behind the FL techniques I proposed an approach while using the PhysioNet 2020 datasets.

It's structure is explained in figure [4.17].

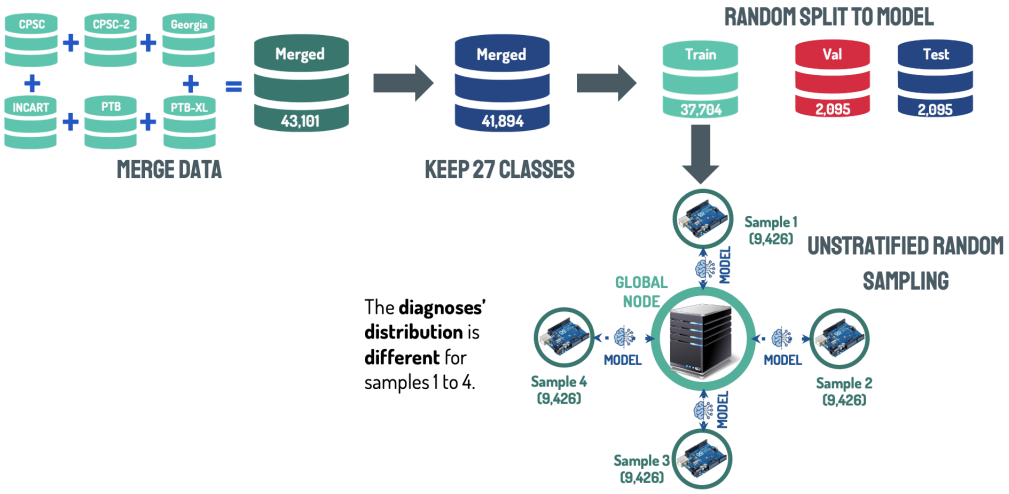


Figure 4.17. Proposed approach

In the above approach, I combined all six datasets (additional information here 3.2) into a single dataset including 43,101 recordings. I came up with 41,894 after utilizing the methodology described in figure 4.17 (filtering out the non-representative classes). After that, I divided the data into train, validation, and test datasets at random. Later, I used an Unstratified random sampling (with replacement) method to acquire four separate samples from the original data. As a result, each sample has a varied distribution of labels (diagnoses).

4.4 Model conversion process

As depicted in 3.2 , most of teams used **Pytorch** during the challenge implementation. As I am going to use Tensorflow as a primary framework and later Tensorflow Lite for using them on mobile phones or IoT devices. Before going to the conversion process itself let's look at the fundamentals of Pytorch, Tensorflow and Tensorflow Lite.

4.4.1 Pytorch:

PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Meta AI. It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

Several pieces of deep learning software are built on top of PyTorch, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, and Catalyst.

Features:

1. Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU)
2. Deep neural networks built on a tape-based automatic differentiation system.

4.4.2 Tensorflow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on the training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2001. TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

Features:

1. *AutoDifferentiation* is the process of automatically calculating the gradient vector of a model concerning each of its parameters.

2. TensorFlow includes an *eager execution* mode, which means that operations are evaluated immediately as opposed to being added to a computational graph which is executed later.

3. In both eager and graph executions, TensorFlow provides an API for distributing computation across multiple devices with various distribution strategies.

4. To train and assess models, TensorFlow provides a set of **loss** functions (also known as cost functions).

There are several other features Tensorflow framework provides in which Tensorflow Lite is one of them.

4.4.3 Tensorflow Lite:

TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices.

Features:

1. Optimized for on-device machine learning, by addressing 5 key constraints: latency (there's no round-trip to a server), privacy (no personal data leaves the device), connectivity (internet connectivity is not required), size (reduced model and binary size) and power consumption (efficient inference and a lack of network connections).

2. Multiple platform support, covering Android and iOS devices, embedded Linux, and microcontrollers.

3. Diverse language support, which includes Java, Swift, Objective-C, C++, and Python.

4. High performance, with hardware acceleration and model optimization.

5. End-to-end for common machine learning tasks such as image classification, object detection, pose estimation, question answering, text classification, etc. on multiple platforms.

4.4.4 Conversion steps:

As we have seen above definitions and features of different frameworks. Let's now look at how we can convert the Pytorch model to TensorFlow lite.

1. First we train the model using a dataset or get a trained model from any challenges team as most of them already stored them in .pt or .pth format. We can utilize open source **Netron** viewer to look at model.

2. After the model is trained we will convert .pt or .pth file into .onnx format by utilizing *torch.onnx.export* function from PyTorch library. We can also use *onnxruntime* package to perform inference on the converted onnx model. *ONNX* is an open format built to represent machine learning models. *ONNX* defines a common set of operators - the building blocks of machine learning and deep learning models - and a common file format to enable AI developers to use models with a variety of frameworks, tools, runtimes, and compilers.

3. After we have model in onnx format we can utilize *onnx-tensorflow* package to convert onnx model to *protobuf* as .pb file. The *protbuf* file contains the graph definition as well as the weights of the model. Thus, a .pb file is all you need to be able to run a given trained model. It is important to note that the .pb file is a frozen version of the model.

   ```
   onnx-tf convert -i /path/to/input.onnx -o /path/to/output
   ```

   using above command after installing the package should convert the model to .pb format

4. After that we can directly load the model using *tf.lite.TFLiteConverter.from_saved _model* function and following the simple code as described `https://www.tensorflow.org/lite/guide/ops_select#convert_a_model`

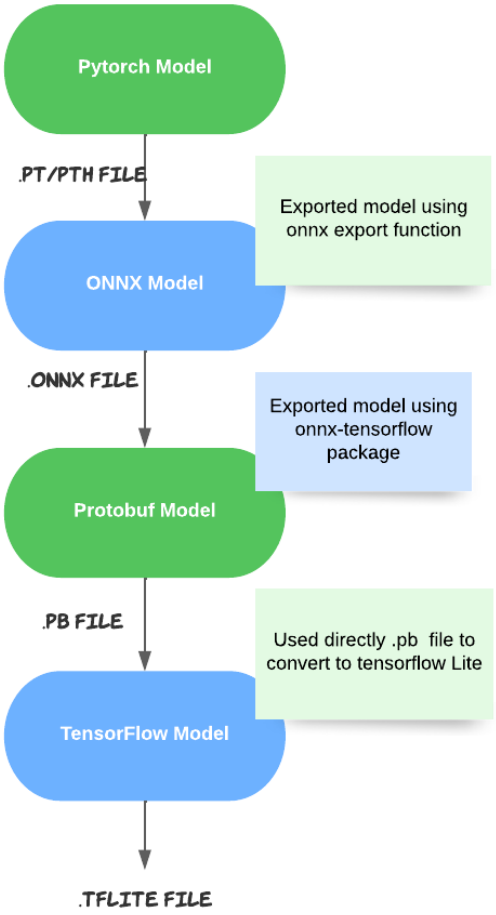


Figure 4.18. Model conversion process

Following the above process or subset of the above process, I was able to convert our DNN ROS model[4.1.5], team 2 model[4.1.5] and team 20 [4.1.5].

Chapter 5

ECG Arrhythmias classification

5.1 Centralized Learning

To get an understanding of the best performances achievable with the mentioned dataset I implemented Centralized (or traditional) Learning. The latter means that I applied the analytical tools mentioned in 4 over the complete dataset, without dividing it into clients. Then the main processes and results are summarized in the following phases.

5.1.1 Data wrangling

As mentioned in 4.1.1, the data wrangling process is usually the first step when dealing with a data-oriented problem. In this case, I placed the data in a Google Drive folder after downloading it from the official competition's website [25]. Afterwards, using Google Colaboratory I extracted and organized the information in Python. The representation of the ECG along the 12-leads can be examined in Figure 5.1.

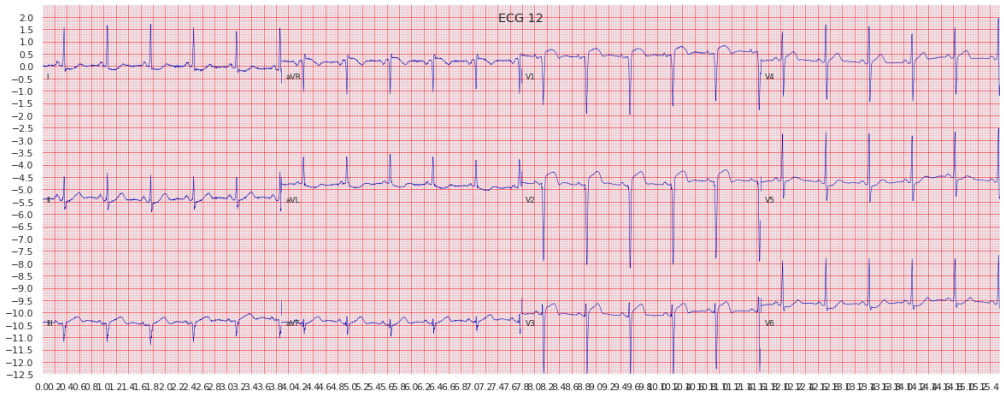


Figure 5.1. 12-lead ECG for recording S0033 of PTB database

Then, the whole data (all the databases) and the features mentioned in 4.1.2 was calculated. That process took almost 1 hour to run and used the default configuration of Google Colab. Besides, for each recording, I selected the first diagnosis (arrhythmia) that appeared as the label to be predicted. The latter process ran in about 2 minutes.

5.1.2 EDA

Once the big dataset was loaded, it contained a total of 43,101 recordings and about 764 variables. From the latter, 650 features got created containing 14 competition-provided features and 636 were spectral features. There were remaining 3 corresponded to the id of the recording, the database that it belongs to and the label (response variable to be predicted). Then, the first analysis needed was to examine if the features contained any missing values. Using the function `bar` from the `missingno` library, I managed to explore the missing values. In figure 5.2 are depicted only the 50 first features' missing counts and percentages.

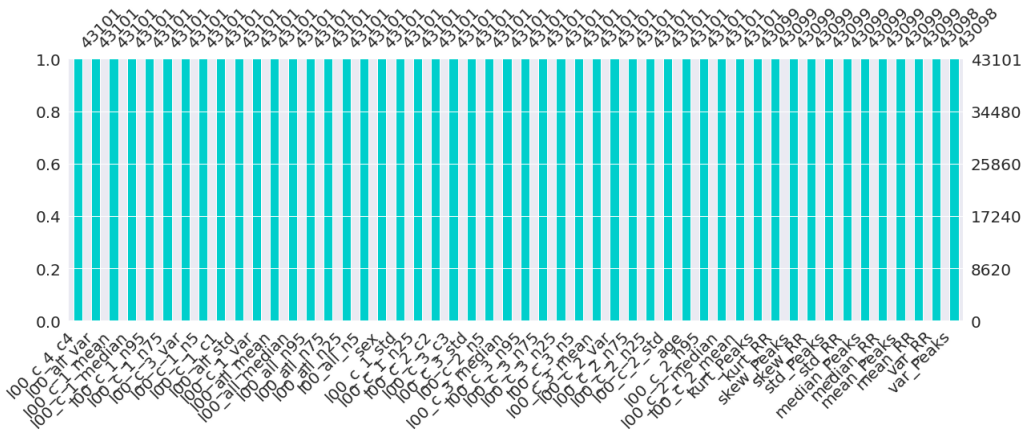


Figure 5.2. Missing counts and percentage for 50 features of the complete dataset

As shown in the previous chart, the missing percentage is considerably small (less than 0.1%). For that reason, I decided to impute those missing values by using the mean (average) of each attribute.

The second crucial aspect to investigate was the distribution of the response variable. Then, within the plot 5.3 it is shown the absolute count of each diagnosis in the dataset.

Figure 5.3. Label distribution for the complete dataset

As evidenced in the previous chart, there are too many arrhythmias that don't have big participation. That can lead to problems when trying to infer the predicted class of a recording since there were not enough cases to learn the classifiers properly. That is why those diagnoses with participation smaller than 150 records were discarded from the analysis. With the previous filter, the selected data to work with got a size of 41,894 recordings distributed as shown in figure 5.4.

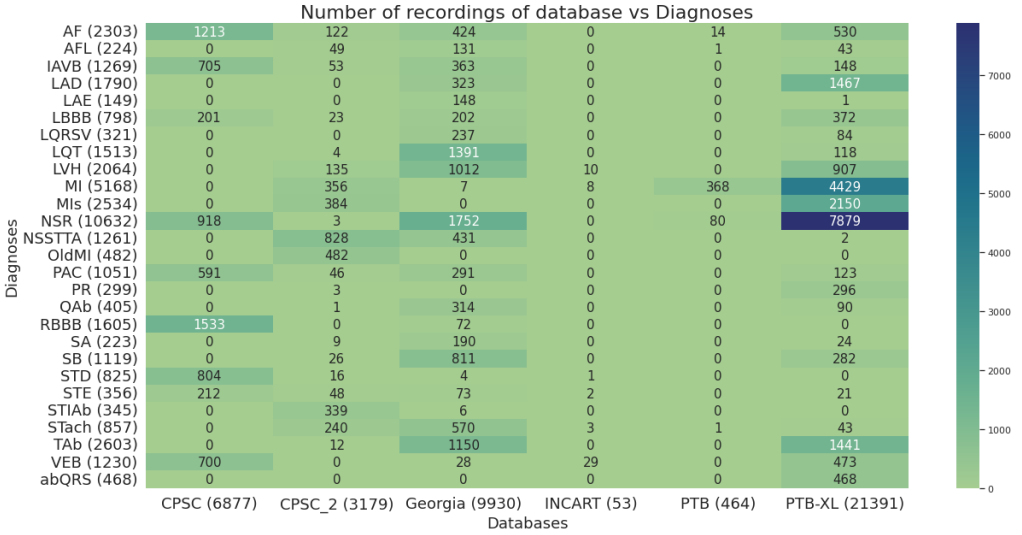


Figure 5.4. Number of recordings for each diagnosis by database for the filtered data

Besides, the final distribution of the labels ended up as shown in figure 5.5. As expected, the most common diagnosis was Normal Sinus Rhythm (NSR), which is the normal status for an ECG. In addition. the arrhythmia with one of the smallest participation turned out to be Sinus Arrhythmia (SA).

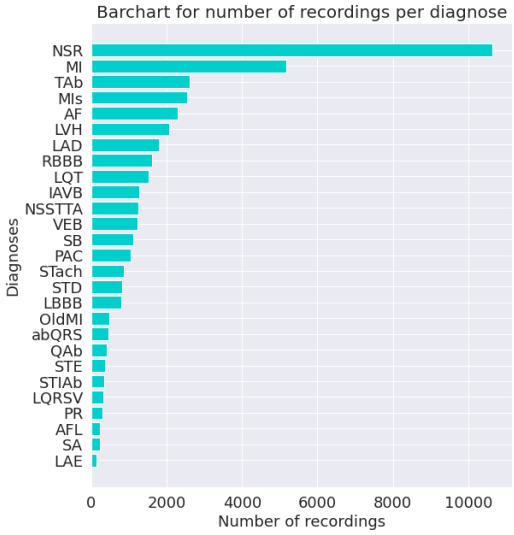


Figure 5.5. Label distribution for the filtered dataset

5.1.3 Feature Selection and normalization

In an analytical context, having a huge amount is a double-edged sword. On the one hand, the more information exists to predict phenomena, the better. On the other hand, the computational time required to process too much information may

lead to training times that are not affordable. Regarding the latter, I decided to perform a feature selection step to determine the most important features to predict arrhythmias.

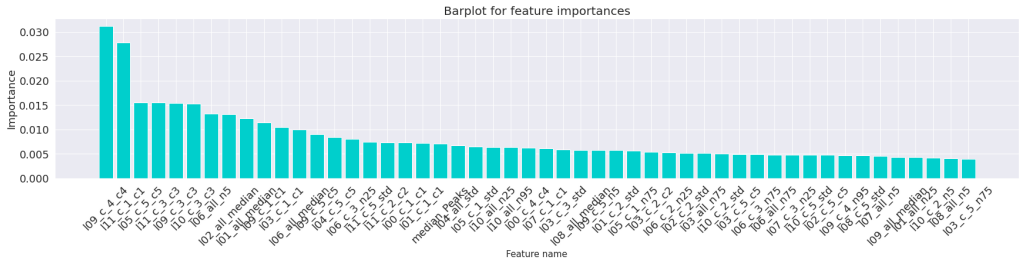


Figure 5.6. Feature importance from XG-Boost algorithm (only the 50 best)

The bar-plot in figure 5.6 depicted the most important features to predict the classes obtained employing the XG-Boost method. The latter provides an automatic ranking of the most relevant features to classifier the ECGs. After having the importance of features of all 650 features, the next step was to try to reduce the unimportant features and run the model. During the various trial run and keeping the F1-Score as the primary metric reducing ranked features from 150 -100 F1-score went down from 0.56 to 0.52 and keeping it on 120 features it goes back to 0.56. So, in the end, 120 features got selected which leads to good enough accuracy and f1-score compared to the one obtained using all the features. The best features turned out to be the Entropy for leads: 9, 11, 10; the percentile 5% for lead 6; and the median for leads 1, 2.

As an additional tool to enhance the performance of the models, there was an implementation of features **normalization**. In this case, I tried three different techniques to transform the features to the same scale. The approaches tried were provided by the `sklearn` library in Python. Those are: `StandardScaler`, `MaxMinScaler` and `RobustScaler`. In the end, the scenario that provided the best results was using `RobustScaler`.

$$X_{scale} = \frac{x_i - x_{med}}{x_{75} - x_{25}} \tag{5.1}$$

`RobustScaler` [5.1] uses statistics that was resistant to outliers to scale features. The median was removed, and the data were scaled according to the quantile range (defaults to IQR: Interquartile Range). The interquartile range (IQR) is the distance between the first and third quartiles (25th and 3rd quantiles) (75th quantile). It is worth mentioning that centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the given dataset.

5.1.4 Balancing classes (arrhythmias)

As depicted in 5.5, the diagnoses have a imbalanced characteristic. The latter means that each category has different participation in the data. That could represent a problem in the performance of the classifiers that will be proposed.

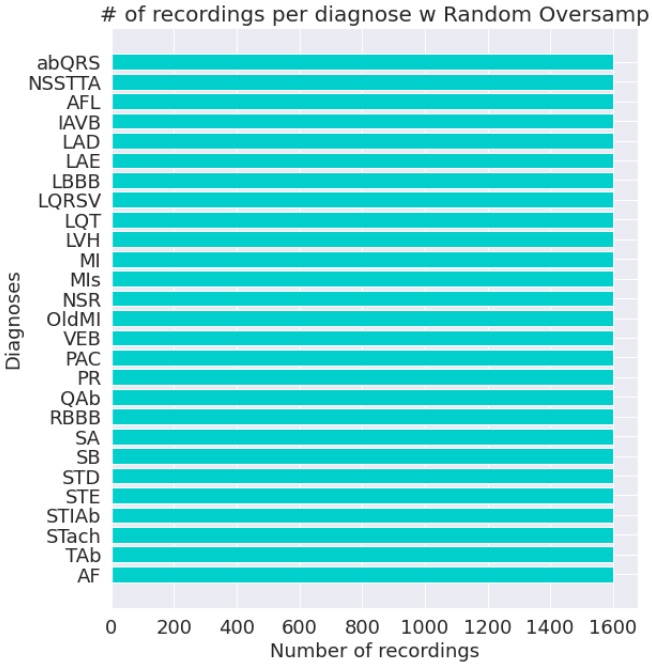


Figure 5.7. Number of recordings for each diagnosis by database for the ROS oversampled data

Then, two oversampling methods were proposed to deal with the imbalance issue. The first one is called **Random Oversampling (ROS)**. In the latter, the minority classes are replicated together with their features. Besides, a down-sampling was applied to have several recordings similar to the filtered dataset. In the end, the ROS dataset had 43,200 recordings. And as depicted in figure 5.7, the distribution of the labels is much more similar among the arrhythmia categories.

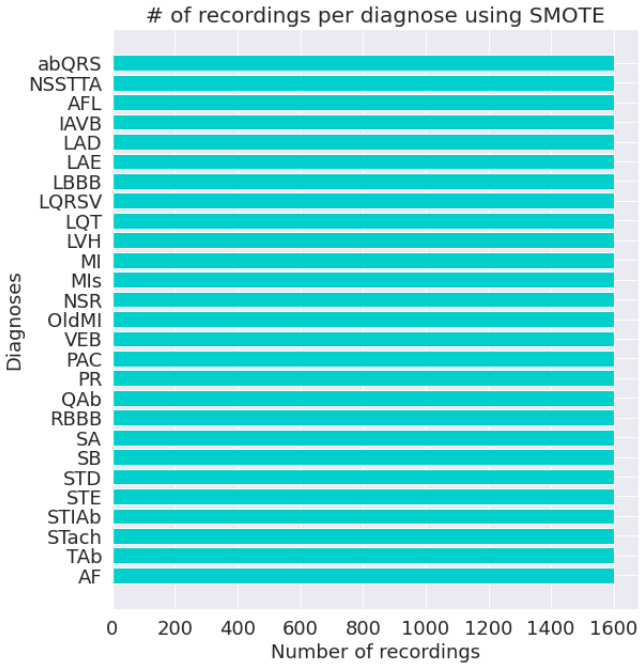


Figure 5.8. Number of recordings for each diagnosis by database for the SMOTE oversampled data

The second oversampling technique used was SMOTE (SMT) 4.1.4. In the end, the SMOTE dataset had also 43,200 recordings. And as shown in figure 5.8, the distribution of the labels is also similar among the arrhythmia categories.

5.1.5 Fitted models and results

With the previous pre-processing applied over the data, the following step was to adjust some Machine Learning models to the ECG's arrhythmias. During this step also other scenarios and considerations were employed [56]. A detailed explanation of the outlines is discussed in table 5.1.

Characteristic	Scenarios	Best approach
Data Split	%Train-%Validation-%Test: Option 1: 60%-20%-20% Option 2: 70%-10%-10% Option 3: 80%-10%-10% Option 4: 90%-5%-5%	Option 4: 90%-5%-5%
Features normalization	Option 1: MinMaxScaler Option 2: StandardScaler Option 3: RobustScaler	Option 3: RobustScaler
Sampling rate	Option 1: 257Hz Option 2: 500Hz	Option 1: 257Hz
Features employed	Option 1: Baseline features Option 2: Baseline features + Spectral features	Option 2: Baseline features + Spectral features

Table 5.1. Scenarios tried during modelling phase

In the previous table are depicted the best approaches that managed to get the best performances (in terms of Accuracy, Recall, Precision, F1-Score and Overall Score). Moreover, in figure 5.9 is shown the detailed behaviour of each one of the algorithms employed.

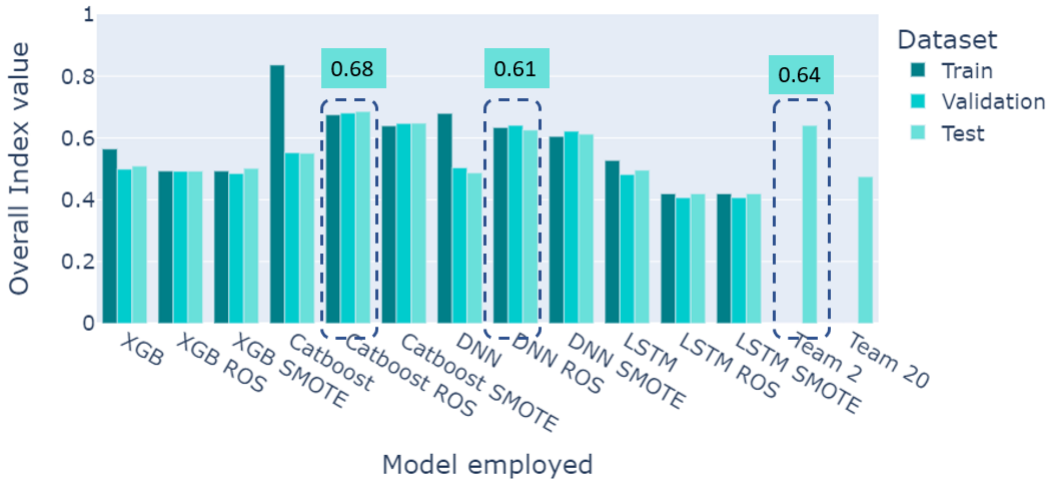


Figure 5.9. Overall Index for methods employed in Centralized Learning (CL)

Within the previous results, it is easy to realize that the champion model is the one that applies Catboost over the ROS data. The latter obtained an overall index close to 0.68. The competition team which ranked 2 is at the second position having an overall index of 0.64. This model has been trained over the above 5.1

scenario having trained data as 90%, 5% validation data and 5% for testing and the same pattern has been followed on team 20 analysis. Nevertheless, the Deep Neural Network (DNN) applied over the ROS data has a close behaviour, having a mentioned metric of 0.61. Finally, LSTM and XG-Boost do not perform that well compared to the other models. An additional point to mention is that both Catboost and DNN used over the ROS dataset do not show any sign of overfitting since the metrics for train, validation and test sets are almost the same. As for team 2 and team 20 in the above figure train and validation overall index is not shown because they followed their pattern keeping in mind the competition score during training and validation.

Figure 5.10. Accuracy for methods employed in Centralized Learning (CL)

Figure 5.11. F1-Score for methods employed in Centralized Learning (CL)

A similar analysis can be derived from figures 5.10 and 5.11, where the Accuracy and F1-Score are shown. In terms of Accuracy, Catboost got 0.67, team 2 got 0.64 and DNN got 0.61 while applied over the ROS dataset. It is important to clarify that using Accuracy is not the best metric in this dataset since the labels are heavily unbalanced. For that reason, it is better to use the F1-Score. which is more robust to the imbalanced datasets. Then, in terms of F1-Score, Catboost obtained a 0.67, the team 2 model got 0.63 and DNN 0.62.

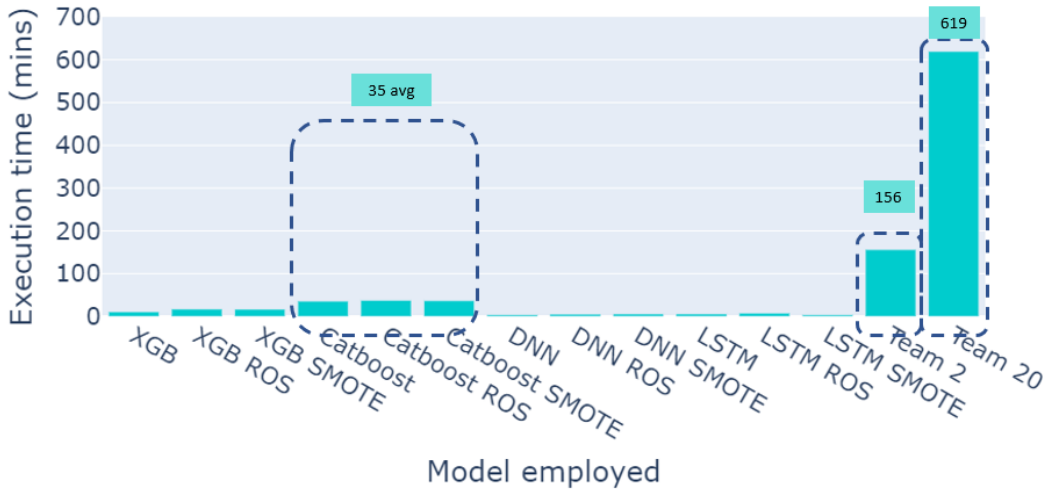


Figure 5.12. Execution times for the methods used in CL

Finally, a measurement of the time taken to train for each model was included.

As shown in figure 5.12, Team 20 model took almost 619 minutes, team 2 took 156 minutes and Catboost took almost 35 minutes to run. On the other hand, DNN and LSTM are algorithms that are employed a few times in training (close to 4 minutes on average). Finally, Catboost is the slowest method, although it generates the best results. In the counter position, DNN is the fastest method, and the performance is NOT quite different from Catboost and the team 2 model. Team 2 and team 20 models took this much time because they are end-to-end models instead of the feature-based model. Although I have done some experiments on team 2 which will be explained below.

5.2 Conversion to Tensorflow Lite

We can convert potentially any model that is built using TensorFlow core libraries and tools. Once you've built a model with TensorFlow core, you can convert it to a smaller, more efficient ML model format called a TensorFlow Lite. Also, there are other advanced steps related to it if your model is built using TensorFlow core libraries by looking at the steps here 4.18 we can convert models that are built using either Pytorch or Matlab.

5.2.1 Model 1(XGBoost) and Model 2(Catboost) conversion

As both of XGBoost and Catboost are not built using Tensorflow even there is no direct conversion available I opted out to use trained models built on top of CatBoost and XGBoost. Just to keep in mind that both of these libraries somehow support directly or indirectly the iOS platform but not the android. After conversion, I am going to use them directly on both iOS/Android to see how they perform in real world scenarios.

5.2.2 Model 3(LSTM)

LSTM model is built on top of Tensorflow so it was convertible to TensorFlow lite. If we look at LSTM models of simple LSTM, LSTM ROS, LSTM SMOTE, a better F1-score and overall index are achieved by simple LSTM model with a 0.49 overall index and 0.47 F1-score. LSTM had one problem during the conversion step it had two subgraphs that are not supported by TensorFlow lite models. To overcome this problem **TFLite LSTM op** was used and in the end, we get the graph that had only one subgraph.

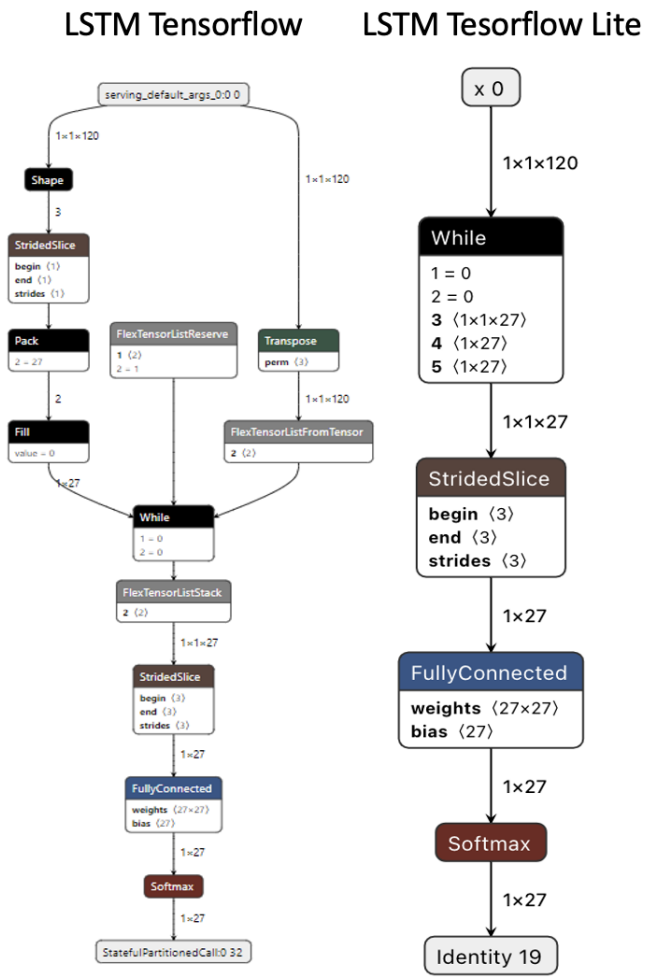


Figure 5.13. Tensorflow LSTM model's graph to Tensorflow Lite model's graph comparison

As you can see in 5.13, LSTM Tensorflow original model and TensorFlow Lite model after the conversion step has taken place using TFLite LSTM op.

5.2.3 Model 4(DNN)

As DNN model was also built using Tensorflow with 4 fully connected layers 3 with 500 neurons output in each and relu as activation function with output. In the fourth layer, it has to output 27 classes of arrhythmias with probabilities so it uses Softmax in the last layer. To look at its graph in the TensorFlow lite model it is not that interesting and the only interesting thing to note is that TensorFlow lite as I have converted only supports one input at a time and Tensorflow one can support batches of inputs as well. As I have to use the TensorFlow Lite version on mobile so most probably it will be given an input to classify one at a time.

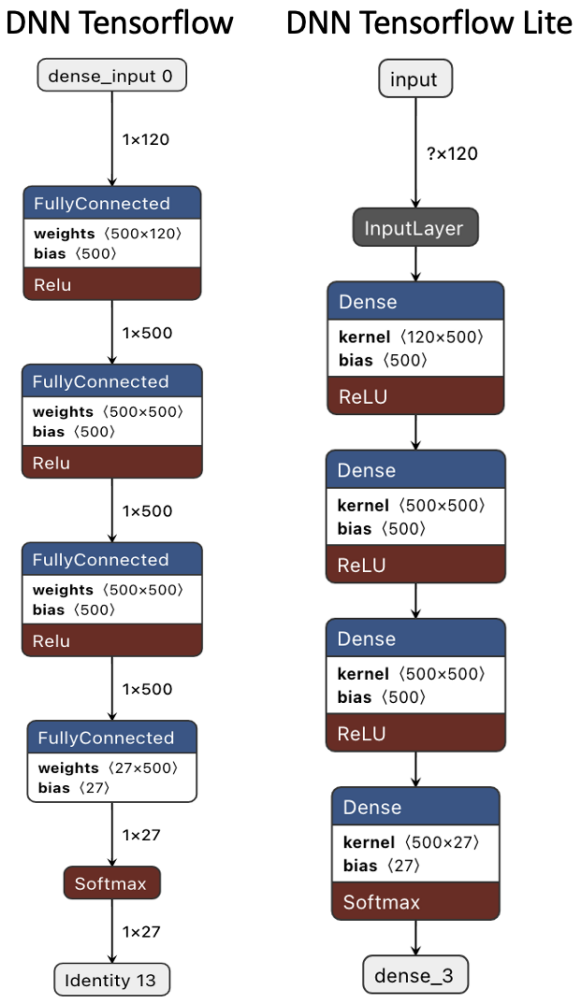


Figure 5.14. Tensorflow DNN model's graph to Tensorflow Lite model's graph comparison

5.2.4 Model 5 - Challenge Team 2

As challenge 2 team was using Pytorch instead of Tensorflow as a primary framework library to build the model. I have to convert the model following all the steps mentioned in 4.18. So, the first trained model .pt file is converted to ONNX format and gets tested, then it got converted to .pb file and at the end when .pb gets imported to TensorFlow it got directly converted to Tensorflow lite. As challenge team 2 was using an ensemble approach where they were using 5 models which got trained over a shuffled dataset for training and validation. We will look at the first model in .pt format, how it looks and then converted the model in Tensorflow Lite as a comparison. Also, this model length is too big to compare here, I will be using the first few layers as a comparison here.

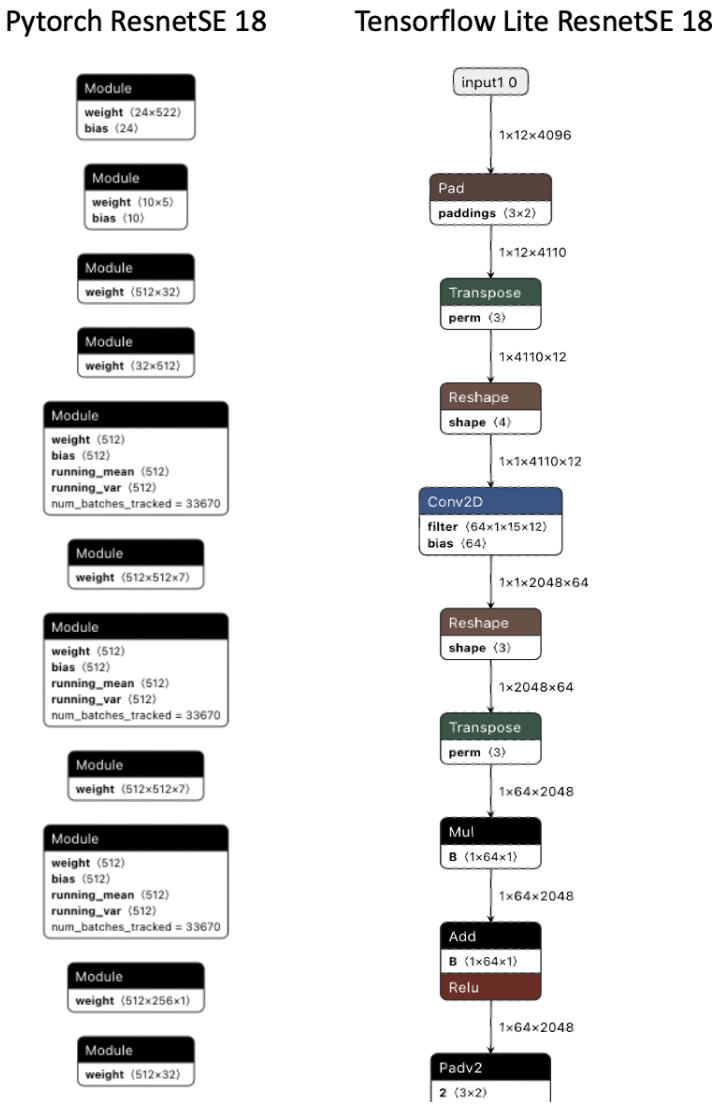


Figure 5.15. Pytorch ResnetSE 18 model's graph to Tensorflow Lite model's graph comparison

As we can see in 5.15 during conversion to TensorFlow Lite there are too many operations that get performed including pad, Transpose, Reshape, Mul, etc. So, we can conclude that in the conversion process to Tensorflow Lite there are additional operations get added to model graphs to support some of the complex operations.

5.2.5 Model 6 - Challenge Team 20

As Challenge team 20, worked directly on the Tensorflow framework library it was easy to convert the model to Tensorflow Lite directly. Let's look at the original model and converted model comparison side by side. It is also important to mention

that as they were training 10 different models using a shuffled dataset and I choose the one for comparison which was performing better in Accuracy, F1-Score and the overall index. It is important to note that the difference between these models was really small in terms of metrics as mentioned before. Model graph length is too big to paste it here so I will be comparing the first few layers of it as in the case of Model 5. 5.2.4

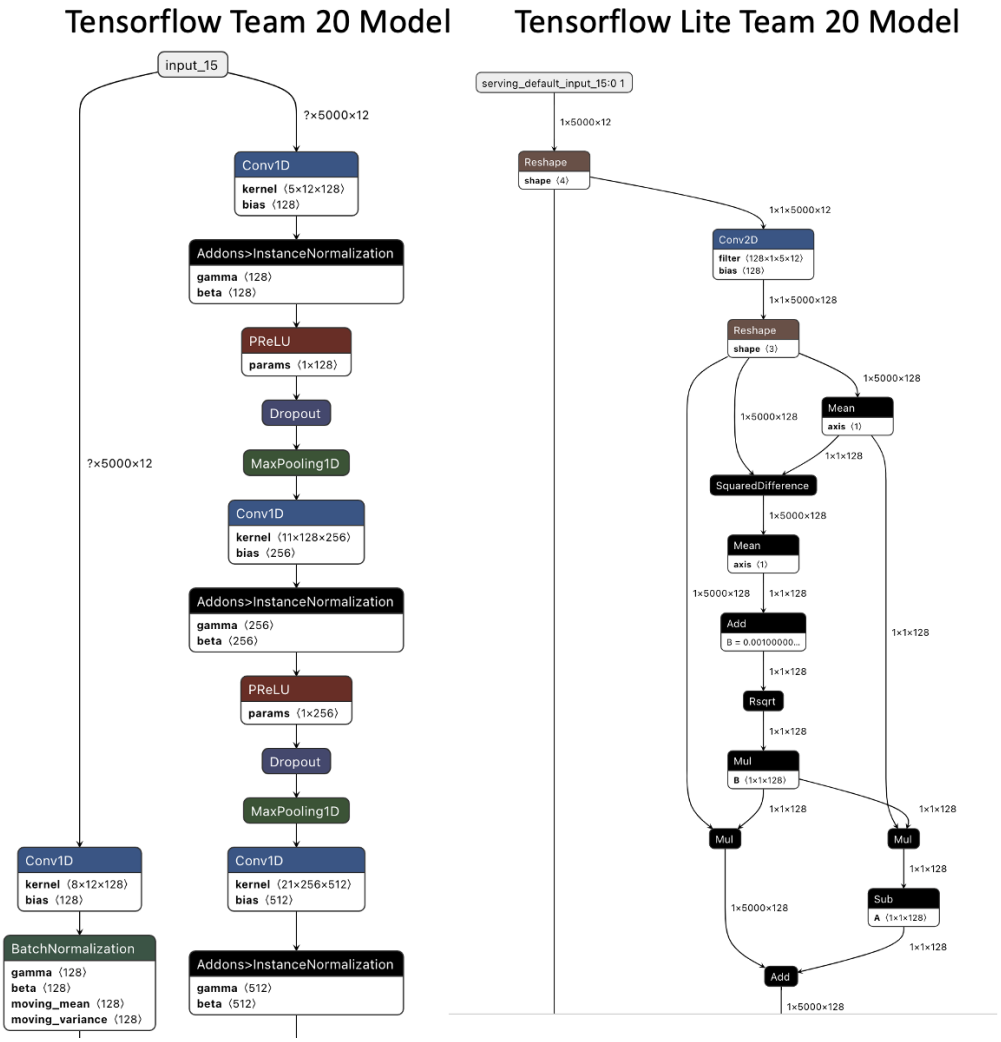
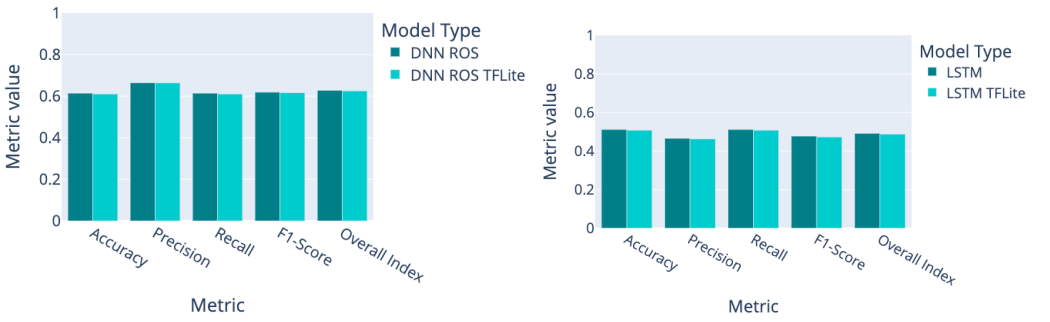


Figure 5.16. Tensorflow Team 20 model's graph to Tensorflow Lite model's graph comparison

Looking at the above figure 5.16, we can clearly see that in the TensorFlow Lite graph there are multiple other operations that we haven't seen before like Sub, SquaredDifference, Mean, Rsqrt etc. It is evident all the complex operations get converted to simpler operations in the TensorFlow Lite format of the model.

5.2.6 Metrics of models

As written previously, Catboost and XGBoost 5.2.1 are not convertible directly to TensorFlow Lite. Besides, those other models were convertible and it was only worth looking at the models that performed well in terms of accuracy, f1-score and overall index because they will be later used for detecting arrhythmias in mobile applications. So, DNN ROS with an overall index of 0.62, LSTM with an overall index of 0.49, Team 2 model with an overall index of 0.64 and Team 20 model(CNN+Rule Based) with an overall index of 0.47 was chosen for the below analysis.



(a) DNN ROS **(b)** LSTM

(c) Team 2 **(d)** Team 20

Figure 5.17. Metrics of models comparing side by side with original model and TF Lite version of the model

If we look at the above figures we can clearly see a little drop in every metric after the model is converted to TensorFlow Lite. As, I didn't want the models to be quantised more so that their accuracy, and F1-Score decrease so I opted for TensorFlow Lite default conversion operations.

Now if we move toward looking at how much time it takes to get output from the model either before converting or after converting. There is huge difference when a model is converted and we wanted to get the output from the model. It takes

slightly and in the case of the hefty model, it takes a lot more time. In real world scenario as we will not be using batching, or multiple inputs to get the output from the model because all above-converted models only accept one input at a time.

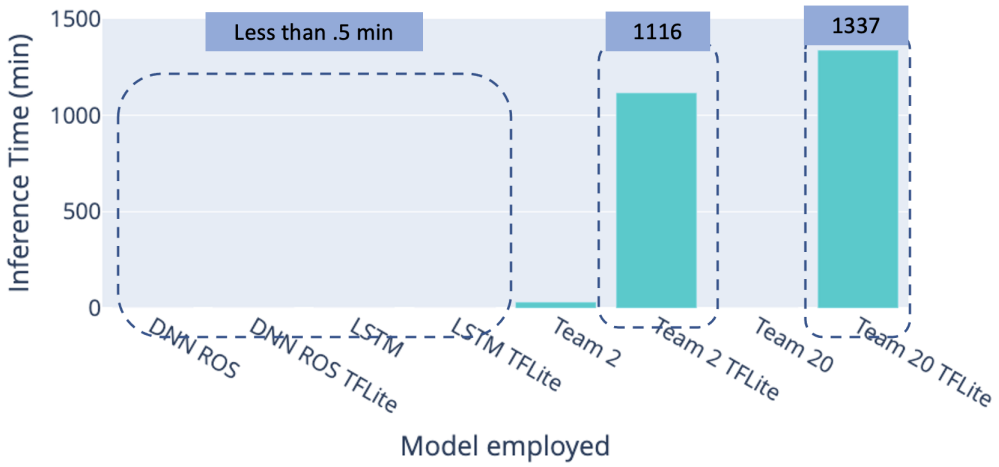


Figure 5.18. Inference Time in minutes of original and converted models

If we look at the above 5.18 figure we can clearly see that LSTM and DNN ROS models are way faster than Team 2 and Team 2. LSTM and DNN ROS models even after conversion is taking less than 0.30 sec to execute. To put it more precisely original DNN on a test dataset with 2095 features takes 19 seconds to get outputs and .13 seconds for inference in the case of a converted model which is even less than the original model. If we look at LSTM it does the same 13 seconds for the original model and .067 seconds on the converted model. This means that somehow both DNN and LSTM have to perform fewer operations and get lighter and more efficient after conversion. While on the other hand Team 2 and Team 20 models take significantly too much time to output the results. More precisely, in the case of the Team 2 model, it takes 1826 seconds to output 2095 results and 66960 seconds in the case of the converted model which raises concerns because it is significantly slower than the original model. As in the case of team 20 inference on the original model was significantly smaller at 9.4 seconds and inference on the converted model was huge slower at 80232 seconds. It may be due to CPU or GPU inference, I didn't get a chance to look at it.

As you can see from 5.2, the smallest model is LSTM and it even gets smaller when converted to TensorFlow lite. Team #20 model is really huge and it gets converted to 11.6MB. Maybe because of that it takes time during inference.

Size of Models		
Model's Name	Original Size	Converted size
DNN ROS	2.2MB	569KB
LSTM	**82KB**	**29KB**
TEAM#2	33.8MB	9.6MB
TEAM#20	138.8MB	11.6MB

Table 5.2. Comparison of model size from original to converted models

5.2.7 Models Deployment

As part of the model testing on mobile devices after conversion. LSTM, DNN ROS and Team #2 model have been deployed on the mobile devices while building a small demo application using flutter. Flutter is an open-source UI software development kit created by Google. As flutter supports Tensorflow Lite models only Catboost, XGBoost are not deployed, as well as the Team #20 model because the accuracy, F1-Score of the team #20 model is very less.

In the demo application, two main packages has been used tflite_flutter and csv. *csv* package has been used to read csv files and *tflite_flutter* is used to inference converted models.

DNN ROS and LSTM

For DNN ROS and LSTM, I saved the features of the test dataset 5% after normalizing and standardizing as explained above 5.1.3 in the csv file. This csv file is used directly for inputs to the models. Then as both models are trained to classify single arrhythmia. The output of the model is based on classifying 27 arrhythmias so whichever arrhythmia type is at the highest probability is shown as predicted and then two more types according to the highest probabilities are shown in 5.21

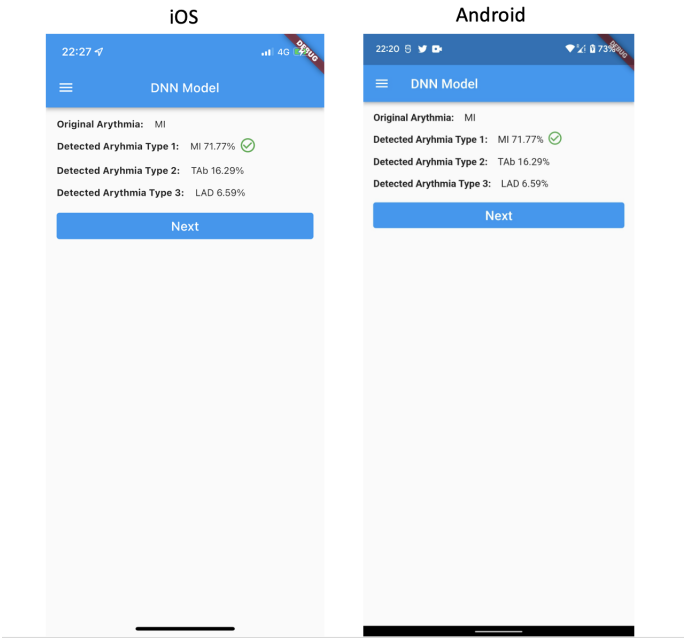


Figure 5.19. iOS and Android DNN model classification example

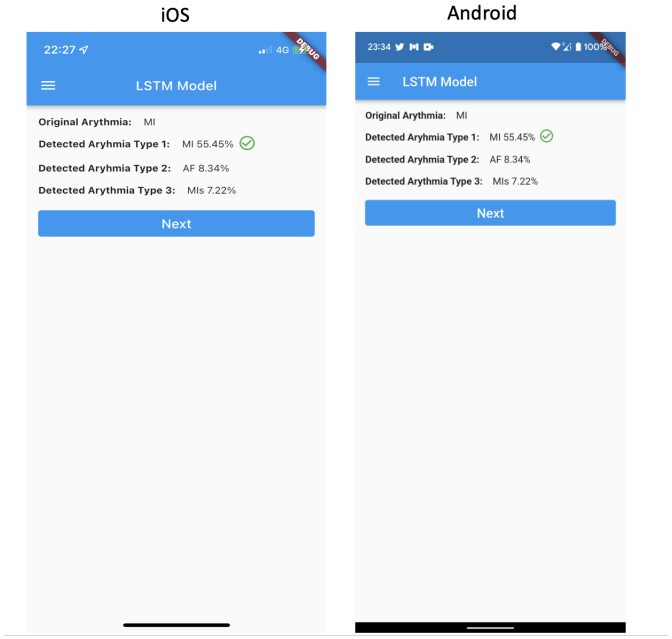


Figure 5.20. iOS and Android LSTM model classification example

Competence Team # 2

For competence team # 2, as they were using end to end model to classify 12-lead ECG, so for each input after pre-processing was saved as one csv file instead .mat file in the original dataset and age, sex and dx is saved separately as well in another

csv file to provide it to the model. As competence team # 2 was using ensemble model, combining 5 of them to output the results for final classification there have been post-processing done on the mobile application as it was in original source code provided by 4.9. If we look closely their model is doing multi-class classification instead of predicting one arrhythmia at a time because a patient can have more than one arrhythmia. Also, during prediction probabilities are divided into a combination of the percentage for how many arrhythmias it predicts. For example, if the model is predicting two different arrhythmias based on 12-lead ECG it will make the whole percentage as 200% instead of 100%.

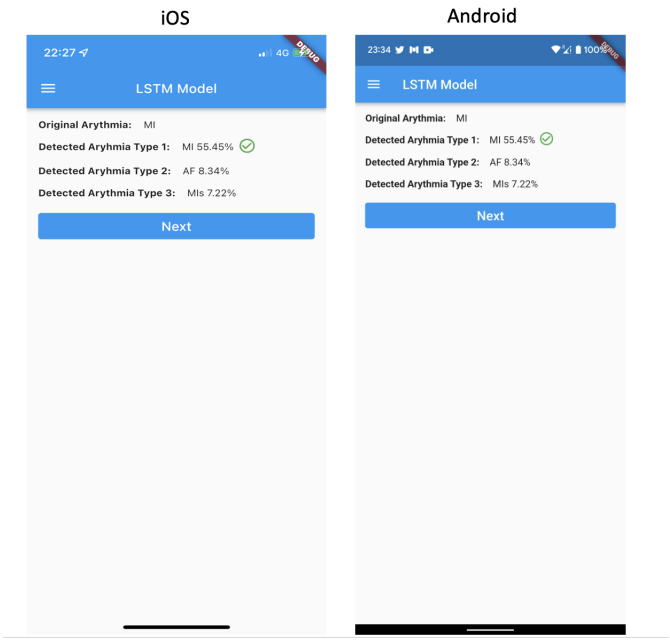


Figure 5.21. iOS and Android Team # 2 model classification example

5.3 Federated Learning

Leveraged on the Centralized Learning method mentioned, it is time to immerse in the Federated Learning (FL) approach executed for this ECG dataset. With the CL it was possible to get an overall performance with the best technique and scenarios to be applied. To deal with the FL proposed IID approach has been chosen for analysis.

5.3.1 IID approach

This approach was based on the idea of using the whole dataset containing 41,894 registers and dividing it into 4 different datasets (local nodes). As a parenthesis, the

decision of the number of local nodes was based on selecting at least 30 diagnoses for the rarest arrhythmia. Due to the stratified random splitting, the mentioned data in each local node (or client) will be Independent and Identically Distributed (IID) 4.3.4. This scenario is not completely realistic since usually the ECGs shared in multiple devices are Non-IID. Nevertheless, it is worth trying and seeing how the FL approach will perform over the data.

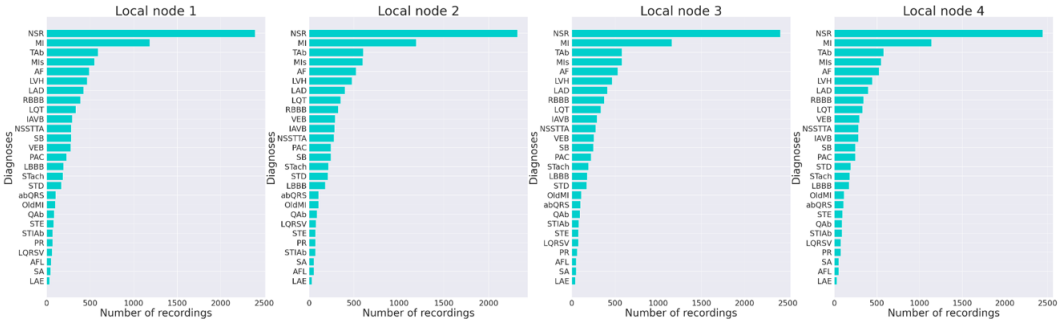


Figure 5.22. Label distribution for the filtered dataset by each local node

As depicted in figure 5.22, the distribution among the 4 local nodes seems IID. The latter means that the diagnoses along the devices will be the same. Besides, each local node contains 9,426 recordings. The same occurs for the ROS and SMOTE datasets when dividing them into 4 clients, as is shown in figures 5.23. Of course, in this case, all the diagnoses have almost the same participation across the nodes, making them IID and balanced.

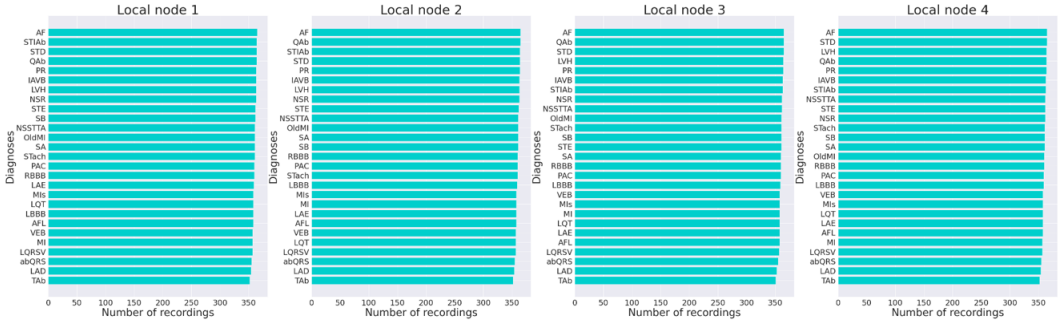


Figure 5.23. Label distribution for the ROS and SMOTE datasets by each local node

Once the four datasets were settled, the modelling part can be performed. As a reminder, in the FL technique, each local node will train a model and later it will send the weights to a global node where the weights are averaged and updated back to each client. Then, the DNN and LSTM methodologies were used to classify the ECG's arrhythmias. It is relevant to highlight that Catboost and XG-Boost were not employed in this step. The former was discarded for its low performance and the latter due to its huge training time. Moreover, in the consulted bibliography, it

is not common to use those ML techniques within the FL.

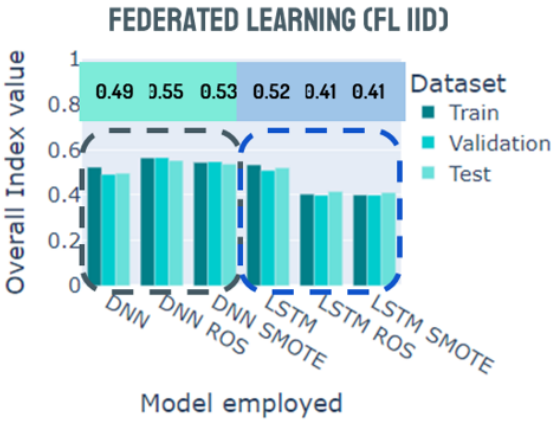


Figure 5.24. Overall Index for methods employed in Federated Learning (FL)

As depicted in figure 5.24, the best results were obtained using the DNN with the ROS datasets. The latter got an overall index of 0.55 in the test set. On the other hand, the best performance for LSTM was obtained with the original data, although it is worst than the DNN ROS model. The latter means that applying oversampling techniques doesn't improve the result of the models in the LSTM approach. But, when using ROS, the performance of the FL model is the most useful compared to all the techniques.

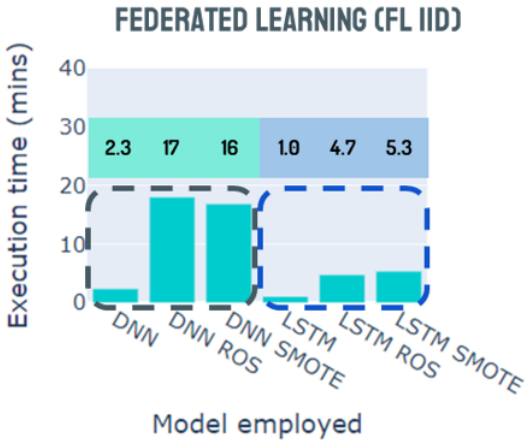


Figure 5.25. Execution times for the methods used in FL IID approach

Analyzing figure 5.25 it is possible to get that the oversampling techniques cause the running time to increase [57]. In detail, the slowest approach ended up being the DNN ROS with 17 minutes. Comparing the results to the CL approach, the DNN method with FL runs faster with similar results. On the other hand, DNN ROS and SMOTE ran slower and with a slower performance than the CL. Finally,

LSTM, in general, has a lower performance than DNN but runs faster.

With FL it is possible to control the performance that the models have in each local node. The latter is relevant to understand if there is some node that is under-performing, compared to the others.

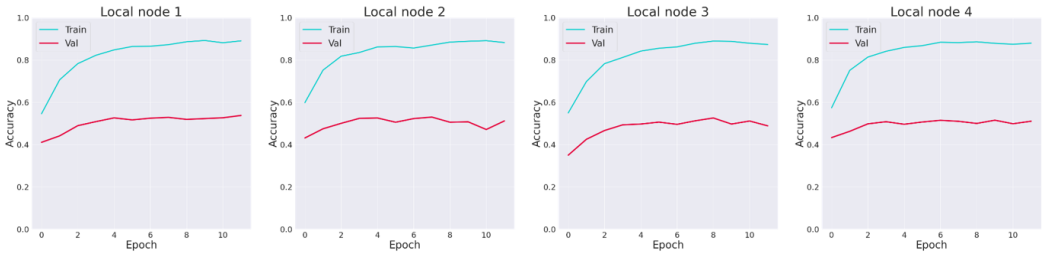


Figure 5.26. Train and Validation accuracy among local nodes for DNN ROS

In figure 5.26 is depicted the behaviour of each local node concerning the accuracy for both the training and validation datasets. In general, the models among the clients have similar behaviour. All of them get stabilized around the 5th epoch. It is important to clarify that the accuracy for train and validation seems considerably far from each other. The latter would mean that the models are over-fitting. Nevertheless, the train sample is an over-sampled data while the validation is a part of the original data, then, it is expected to such behaviour.

In an FL environment, there is a concept called **communication round** (comm round). The latter begins when a model is trained inside each one of the local nodes. Later the weights of the models are passed to the global node to be aggregated there. And finally, the communication round finishes when each local model is updated with the new weights. Then, it is expected that in each comm round the performance of the model increases. Moreover, it should get stable after some trials.

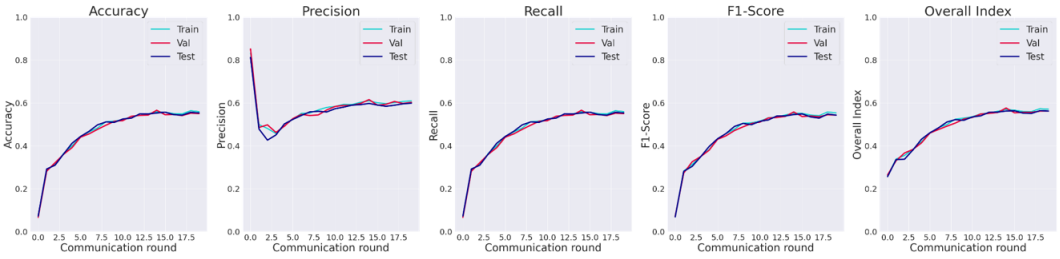


Figure 5.27. Metrics along communication rounds for DNN ROS (the best model)

Figure 5.27 establishes the behaviour of each metric along the communication rounds. As expected, the performance gets stable after the 15th comm round approximately. It is relevant to notice that in the first comm round the Precision starts high and the Recall low. But after some updates both measures get steady. Moreover, the curves for train, validation and test are close. The latter means that

there are no signals of overfitting.

	CL					FL IID				
Method	Accuracy	Precision	Recall	F1-Score	Time	Accuracy	Precision	Recall	F1-Score	Time
Competence Team #2 [15]	0.64	0.64	0.64	0.63	155	-	-	-	-	-
Competence Team #20 [58]	0.47	0.51	0.47	0.44	618	-	-	-	-	-
Inspirational DNN 4.1.5	0.50	0.46	0.50	0.47	**1.4**	-	-	-	-	-
Inspirational LSTM 4.1.5	0.50	0.45	0.50	0.46	2.1	-	-	-	-	-
XG-BOOST	0.52	0.49	0.52	0.49	9.4	-	-	-	-	-
XG-BOOST ROS	0.45	0.58	0.45	0.47	15.7	-	-	-	-	-
XG-BOOST SMOTE	0.48	0.55	0.48	0.49	15.4	-	-	-	-	-
CATBOOST	0.55	0.54	0.55	0.53	34.5	-	-	-	-	-
CATBOOST ROS	**0.67**	**0.72**	**0.67**	**0.67**	36.4	-	-	-	-	-
CATBOOST SMOTE	0.64	0.66	0.64	0.64	35.5	-	-	-	-	-
DNN	0.50	0.47	0.50	0.47	2.8	0.46	0.53	0.45	0.49	2.3
DNN ROS	0.61	0.66	0.61	0.61	3.8	**0.55**	**0.59**	**0.54**	**0.55**	16.3
DNN SMOTE	0.60	0.64	0.60	0.60	4.3	0.52	0.58	0.52	0.51	16.8
LSTM	0.51	0.47	0.51	0.39	4.3	0.48	0.58	0.48	0.52	**1.0**
LSTM ROS	0.38	0.51	0.38	0.39	4.9	0.39	0.48	0.38	0.38	4.7
LSTM SMOTE	0.39	0.49	0.39	0.39	2.7	0.39	0.47	0.39	0.38	5.3

Table 5.3. Metrics for Centralized (CL) and IID Federated Learning (FL -IID) in the test data set. Also included execution time for training in minutes.

The table 5.3 includes a summary of all the techniques implemented with the data in the IID and Centralized (CL) approaches. It is remarkable to mention that Catboost ROS outperform Competence Team # 2 model. DNN ROS is a model that has good enough accuracy and f1-score which can be easily deployed to any small memory device after conversion.

Chapter 6

Conclusions

6.1 Summary

In the end, the categorization of 12-channel ECG arrhythmias was improved by using oversampling (and undersampling) approaches. Random Oversampling (ROS), in particular, performed admirably. The latter showed a considerable increment compared to the model trained with the original data. Although SMOTE performed well in a variety of cases, ROS was chosen as the best strategy due to its ease of implementation.

In conclusion, Catboost with the random oversampling technique was the best model but we cannot deploy that in mobile devices which support both Android/IOS as well as the XGBoost ROS model. DNN ROS, LSTM and Team # 20 model can be deployed to mobile devices easily and perform better with the same accuracy and F1-Score. Any model that is built using Matlab Keras library, Pytorch framework Keras library and Tensorflow Keras library and Tensorflow itself can be converted to Tensorflow Lite and we can see them in action in the above demo 5.2.7. Inferencing big models like Team #2 and Team # 20 models takes a lot of time to get the output after conversion.

As far as Federated Learning is concerned, the DNN trained with ROS (oversampled) data proved to be the best model in the setting of FL. The latter had an Overall Index of 0.55 and an F1-Score of 0.55. Even said, when more than four local nodes were included, utilizing an LSTM with oversampled data produced more consistent results. Then LSTM is the recommended model to implement within this data when the number of nodes is large. On the other hand, DNN with ROS data is the candidate to select when dealing with a small number of clients.

6.2 Future Developments

In terms of future work, converting the competence team # 20 model to utilize it directly in a mobile application can be done easily because it's already converted to Tensorflow Lite. Also, training models of Catboost, XGBoost, competence team # 2 and competence team # 20 in FL settings can be done and looking at their overall index can be good insight for future work. In the end, as we deployed the models in mobile as edge nodes, these can also be deployed to Arduino by utilizing Tensorflow quantization techniques and training them directly in edge devices instead of simulating its behaviour to see how they perform in real settings could be something that requires more resources and can be benefiting in real world performance analysis of these models. Also, after central training, we can utilize transfer learning technique in a federated learning environment where a model has been trained already and it will get trained over a new dataset without sharing patients data keeping in mind privacy concerns.

Bibliography

[1] Alessandro Scirè, Fabrizio Tropeano, Aris Anagnostopoulos, and Ioannis Chatzigiannakis. Fog-computing-based heartbeat detection and arrhythmia classification using machine learning. *Algorithms*, 12(2), 2019.

[2] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletarì, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M. Jorge Cardoso. The future of digital health with federated learning. *npj Digital Medicine*, 3(1), sep 2020.

[3] Qi Xia, Winson Ye, Zeyi Tao, Jindi Wu, and Qun Li. A survey of federated learning for edge computing: Research problems and solutions. *High-Confidence Computing*, 1(1):100008, 2021.

[4] Haftay Gebreslasie Abreha, Mohammad Hayajneh, and Mohamed Adel Serhani. Federated learning in edge computing: A systematic survey. *Sensors*, 22(2), 2022.

[5] Paul A Iaizzo. *Handbook of cardiac anatomy, physiology, and devices, third edition*. Springer International Publishing, January 2015.

[6] Michael Sampson and Anthony Mcgrath. Understanding the ecg. part 1: Anatomy and physiology. *British Journal of Cardiac Nursing*, 10:548–554, 11 2015.

[7] Michael Sampson and Anthony Mcgrath. Understanding the ecg part 2: Ecg basics. *British Journal of Cardiac Nursing*, 10:588–594, 12 2015.

[8] G. Walraven. *Basic Arrhythmias*. Pearson Education, 2014.

[9] Sandra Goldsworthy, Ruth Kleinpell, and Ged Williams. *International Best Practices in Critical Care*. World Federation of Critical Care Nurses„ New York, NY :, 2017.

[10] Erick Perez and matthewreyna. physionetchallenges, 03 2021.

[11] Shah AJ Perez Alday EA, Gu A. Classification of 12-lead ecgs: the physionet/-computing in cardiology challenge 2020. *Physiol Meas*, 2020 Nov 11.

[12] Shenda Hong, Wenrui Zhang, Chenxi Sun, Yuxi Zhou, and Hongyan Li. Practical lessons on 12-lead ecg classification: Meta-analysis of methods from physionet/-computing in cardiology challenge 2020. *Frontiers in Physiology*, 12, 2022.

[13] Erick A Perez Alday, Annie Gu, Amit J Shah, Chad Robichaux, An-Kwok Ian Wong, Chengyu Liu, Feifei Liu, Ali Bahrami Rad, Andoni Elola, Salman Seyedi, Qiao Li, Ashish Sharma, Gari D Clifford, and Matthew A Reyna. Classification of 12-lead ECGs: the PhysioNet/computing in cardiology challenge 2020. *Physiological Measurement*, 41(12):124003, dec 2020.

[14] Annamalai Natarajan, Yale Chang, Sara Mariani, Asif Rahman, Gregory Boverman, Shruti Vij, and Jonathan Rubin. A wide and deep transformer neural network for 12-lead ecg classification. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[15] Zhibin Zhao, Hui Fang, Samuel D Relton, Ruqiang Yan, Yuhong Liu, Zhijing Li, Jing Qin, and David C Wong. Adaptive lead weighted resnet trained with different duration signals for classifying 12-lead ecgs. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[16] Zhaowei Zhu, Han Wang, Tingting Zhao, Yangming Guo, Zhuoyang Xu, Zhuo Liu, Siqi Liu, Xiang Lan, Xingzhi Sun, and Mengling Feng. Classification of cardiac abnormalities from ecg signals using se-resnet. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[17] Jiabo Chen, Tianlong Chen, Bin Xiao, Xiuli Bi, Yongchao Wang, Han Duan, Weisheng Li, Junhui Zhang, and Xu Ma. Se-ecgnet: Multi-scale se-net for multi-lead ecg data. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[18] Najmeh Fayyazifar, Selam Ahderom, David Suter, Andrew Maiorana, and Girish Dwivedi. Impact of neural architecture design on cardiac abnormality classification using 12-lead ecg signals. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[19] Seonwoo Min, Hyun-Soo Choi, Hyeongrok Han, Minji Seo, Jin-Kook Kim, Junsang Park, Sunghoon Jung, Il-Young Oh, Byunghan Lee, and Sungroh Yoon. Bag of tricks for electrocardiogram classification with deep neural networks. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[20] Max N Bos, Rutger R Van De Leur, Jeroen F Vranken, Deepak K Gupta, Pim Van Der Harst, Pieter A Doevendans, and René Van Es. Automated comprehensive interpretation of 12-lead electrocardiograms using pre-trained

exponentially dilated causal convolutional neural networks. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[21] Charilaos Zisou, Andreas Sochopoulos, and Konstantinos Kitsios. Convolutional recurrent neural network and lightgbm ensemble model for 12-lead ecg classification. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[22] Po-Ya Hsu, Po-Han Hsu, Tsung-Han Lee, and Hsin-Li Liu. Multi-label arrhythmia classification from 12-lead electrocardiograms. In *2020 Computing in Cardiology*, pages 1–4, 2020.

[23] PhysioNet. Moody challenge overview, Retrieved on May 31, 2022.

[24] Erick Perez Alday, Annie Gu, Amit Shah, Chad Robichaux, An-Kwok Wong, Chengyu Liu, Feifei Liu, Ali Bahrami Rad, Andoni Elola, Salman Seyedi, Qiao Li, Ashish Sharma, Gari Clifford, and Matthew Reyna. Classification of 12-lead ecgs: the physionet/computing in cardiology challenge 2020. 08 2020.

[25] PhysioNet. Classification of 12-lead ecgs: The physionet/computing in cardiology challenge 2020, Retrieved on June 01, 2022.

[26] Ary Goldberger, Luís Amaral, L. Glass, Shlomo Havlin, J. Hausdorg, Plamen Ivanov, R. Mark, J. Mietus, G. Moody, Chung-Kang Peng, H. Stanley, and Physiotoolkit Physiobank. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *PhysioNet*, 101, 01 2000.

[27] Tye Rattenbury, Joseph M. Hellerstein, Jeffrey Heer, Sean Kandel, and Connor Carreras. *Principles of Data Wrangling: Practical Techniques for Data Preparation*. O'Reilly Media, Inc., 1st edition, 2017.

[28] Ron. Stefanski. Data wrangling in 6 steps: An analyst's guide for creating useful data, Retrieved on May 05, 2022.

[29] Dongdong Zhang. Interpretable deep learning for automatic diagnosis of 12-lead electrocardiogram, 06 2021.

[30] Jozef Surda, Stanislav Lovas, Jozef Pucik, and Milan Jus. Spectral properties of ecg signal. pages 1–5, 05 2007.

[31] Shawhin Talebi. The wavelet transform, 12 2020.

[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002.

[33] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh Chawla. Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 04 2018.

[34] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794, 2016.

[35] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, aug 2020.

[36] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. arXiv, 2017.

[37] Imad Basheer and M.N. Hajmeer. Artificial neural networks: Fundamentals, computing, design, and application. *Journal of microbiological methods*, 43:3–31, 01 2001.

[38] Crescenzio Gallo. *Artificial Neural Networks: tutorial*. 01 2015.

[39] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 07 2009.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[42] Awni Y Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia, and Andrew Y Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature medicine*, 25(1):65–69, 2019.

[43] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[44] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[45] Jiapu Pan and Willis J. Tompkins. A real-time qrs detection algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3):230–236, 1985.

[46] Gürol Canbek, Seref Sagiroglu, Tugba Taskaya Temizel, and Nazife Baykal. Binary classification performance measures/metrics: A comprehensive visualized roadmap to gain new insights. pages 821–826, 10 2017.

[47] Mohammad Hossin and Sulaiman M.N. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining and Knowledge Management Process*, 5:01–11, 03 2015.

[48] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[49] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

[50] Haya Elayan, Moayad Aloqaily, and Mohsen Guizani. Deep federated learning for iot-based decentralized healthcare systems. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 105–109, 2021.

[51] Sadman Sakib, Mostafa M. Fouda, Zubair Md Fadlullah, Khalid Abualsaud, Elias Yaacoub, and Mohsen Guizani. Asynchronous federated learning-based ecg analysis for arrhythmia detection. In *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pages 277–282, 2021.

[52] Mufeng Zhang, Yining Wang, and Tao Luo. Federated learning for arrhythmia detection of non-iid ecg. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pages 1176–1180, 2020.

[53] Arjun Ramesh Kaushik. Types of Federated Learning, Retrieved on May 30, 2022.

[54] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. 2019.

[55] Anna Bogdanova, Nii Attoh-Okine, and Tetsuya Sakurai. Risk and advantages of federated learning for health care data collaboration. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 6:04020031, 09 2020.

[56] Quang Nguyen, Hai-Bang Ly, Ho Lanh, Nadhir Al-Ansari, Hiep Le, Tran Van Quan, Indra Prakash, and Binh Pham. Influence of data splitting on performance of machine learning models in prediction of shear strength of soil. *Mathematical Problems in Engineering*, 2021, 02 2021.

[57] Hadi Jamali-Rad, Mohammad Abdizadeh, and Anuj Singh. Federated learning with taskonomy for non-iid data. arXiv, 2021.

[58] Bjørn-Jostein Singstad and Christian Tronstad. Convolutional neural network and rule-based algorithms for classifying 12-lead ecgs. In *2020 Computing in Cardiology*, pages 1–4, 2020.