Classification of Cardiac Arrhythmias based on Overlapped ECG Image with Lightweight Neural Network

Miss Khaing Su Thway

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Electrical Engineering
Department of Electrical Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2022

การแยกแยะภาวะหัวใจเต้นผิดจังหวะด้วยโครงข่ายประสาทแบบไลต์เวทบนพื้นฐานของภาพซ้อนรูปคลื่นไฟฟ้าหัวใจ

น.ส.เคียง ซู ทเวย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Thesis Title	Classification of Cardiac Arrhythmias based on Overlapped ECG Image with Lightweight Neural Network
By	Miss Khaing Su Thway
Field of Study	Electrical Engineering
Thesis Advisor	Assistant Professor Dr. ARPORN TEERAMONGKONRASMEE
Thesis Co Advisor	Assistant Professor Dr. Pakpum Somboon

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Engineering

..... Dean of the FACULTY OF ENGINEERING

(Professor Dr. SUPOT TEACHAVORASINSKUN)

THESIS COMMITTEE

..... Chairman

(Associate Professor Dr. SUPATANA AUETHAVEKIAT)

..... Thesis Advisor

(Assistant Professor Dr. ARPORN TEERAMONGKONRASMEE)

..... Thesis Co-Advisor

(Assistant Professor Dr. Pakpum Somboon)

..... External Examiner

(Associate Professor Dr. Wisarn Patchoo)

เคียง ซู ทเวย์ : การแยกแยะภาวะหัวใจเต้นผิดจังหวะด้วยโครงข่ายประสาทแบบไลต์เวทบนพื้นฐานของภาพซ้อนรูปคลื่นไฟฟ้าหัวใจ. (Classification of Cardiac Arrhythmias based on Overlapped ECG Image with Lightweight Neural Network) อ.ที่ปรึกษาหลัก : อาภรณ์ ธีรมงคลรัศมี, อ.ที่ปรึกษาร่วม : ภาคภูมิ สมบูรณ์

วิทยานิพนธ์นี้นำเสนอการจำแนกภาวะหัวใจเต้นผิดจังหวะ ได้แก่ ภาวะ AF, NSR, PAC และ PVC โดยใช้โครงข่ายประสาทแบบไลต์เวท ในขั้นตอนของการแปลงสัญญาณคลื่นไฟฟ้าหัวใจ (ECG) เป็นข้อมูลภาพเพื่อเป็นอินพุตของโครงข่ายประสาท วิทยานิพนธ์นี้ได้พัฒนาวิธีการใหม่ คือ การซ้อนรูปคลื่นไฟฟ้าหัวใจ ในขั้นตอนการแปลง สัญญาณคลื่นไฟฟ้าหัวใจความยาว 30 วินาที จะถูกแบ่งเป็นรูปคลื่นตามอัตราการเต้นหัวใจ รูปคลื่นที่แบ่งได้ถูกวางซ้อนภายในขอบเขตของแกน x และ y วิธีการนี้สามารถสร้างภาพที่มีความแตกต่างระหว่างภาวะหัวใจเต้นผิดจังหวะที่สนใจ ในส่วนของโครงข่ายประสาทที่นำมาใช้เป็นแบบไลต์เวทซึ่งได้ถูกออกแบบให้สามารถติดตั้งบนอุปกรณ์พกพาเนื่องจากโครงข่ายใช้พารามิเตอร์ที่มีจำนวนน้อยลง ประสิทธิภาพของโครงข่ายประสาทที่พัฒนาขึ้นได้ถูกทดสอบโดยฐานข้อมูล LTAF ของ PhysioNet ฐานข้อมูลนี้มีจำนวน 84 ชุดข้อมูล ซึ่งแต่ละชุดข้อมูลมีความยาวประมาณ 24 ชั่วโมง ประสิทธิภาพของการจำแนกภาวะหัวใจเต้นผิดจำแนกได้รับการวิเคราะห์โดยใช้ confusion metric และได้ความแม่นยำมากกว่า 98% นอกจากนี้ แอปพลิเคชันของ Android ได้ถูกพัฒนาขึ้นโดยใช้โมเดลการเรียนรู้ของโครงข่ายประสาท ซึ่งเป็นการยืนยันถึงศักยภาพของการประยุกต์ใช้งานโครงข่ายประสาทแบบไลต์เวทกับโทรศัพท์เคลื่อนที่

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

ปีการศึกษา 2565 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ลายมือชื่อ อ.ที่ปรึกษาร่วม .....

6370392021 : MAJOR ELECTRICAL ENGINEERING
KEYWORD: Lightweight Deep Learning Neural Network, ECG, Cardiac Arrhythmias, Atrial Fibrillation AF

Khaing Su Thway : Classification of Cardiac Arrhythmias based on Overlapped ECG Image with Lightweight Neural Network. Advisor: Asst. Prof. Dr. ARPORN TEERAMONGKONRASMEE Co-advisor: Asst. Prof. Dr. Pakpum Somboon

This dissertation presents the classification of different types of cardiac arrhythmias, including Atrial Fibrillation (AF), Normal Sinus Rhythm (NSR), Premature Atrial Contraction (PAC), and Premature Ventricular Contraction (PVC), using a lightweight neural network. A novel ECG data transformation method, referred to as transforming into overlapped ECG images, has been developed and utilized as input images for our neural network. During the transformation process, individual heartbeats within a 30-second time frame are cropped based on heart rate calculations. These resulting beats are then overlapped with respect to the x and y axis limits, generating distinctive images representing different arrhythmia types examined in this study. The lightweight neural networks employed in this research have been designed to be deployable on low-resource mobile devices due to their reduced network architecture parameters,. The performance of the developed neural network is evaluated using the long-term atrial fibrillation (LTAF) database from PhysioNet, which is clinically certified. This database comprises a total of 84 records, which consist of 24-hour duration for each record. The effectiveness of the proposed approaches is analyzed using confusion metrics, yielding an accuracy rate exceeding 98%. Furthermore, a demonstration Android application has been developed based on the trained model of the lightweight neural network, providing proof of concept for the potential applications of deploying lightweight deep learning neural networks on mobile phones.

Field of Study:	Electrical Engineering	Student's Signature .....
Academic Year:	2022	Advisor's Signature .....
		Co-advisor's Signature .....

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my great advisor, Asst. Prof. Dr. Arporn Teeramongkonrasmee for his guidance and patience. If he had not accepted me as his student, I would not have had the chance to study at Chulalongkorn University. He has also supported and made it possible to do my research, especially for my extended semester.

Secondly, I am deeply grateful to my co-advisor, Asst. Prof. Dr. Pakpum Somboon for his insightful comments and suggestions. Without his persistent help, this thesis would not have been materialized. I have learned a lot from my professors to be focused, organized, systematic, and dedicated to work. It was enjoyable to study under their supervision as they have pointed out my weaknesses and never discouraged me for not meeting their expectations. Because of that, I managed to sharpen my skills and deliver my best.

Thirdly, I would like to show my appreciation to Assoc. Prof. Dr. Supatana Auethavekiat and Asst. Prof. Dr. Wisarn Patchoo for being my thesis committee and for their constructive feedback to improve my thesis. Finally, I thank Chulalongkorn University for providing financial support under the graduate scholarship program for ASEAN and Non-ASEAN countries.

Khaing Su Thway

TABLE OF CONTENTS

Page

1 Introduction

1.1 Motivation and Problem Statement

Cardiac arrhythmia is the abnormal rhythm of the heart related to various underlying causes, including electrical conduction abnormalities within the heart, medication side effects, and other underlying health conditions. According to the study [1], it can contribute to the occurrence of sudden cardiac death in 15-20% of the world population. It can be found even in children and healthy people during intensive exercise[2]. Moreover, the recent survey showed that cardiac arrhythmias are correlated with Covid-19 patients due to the cardiac tissue damage from the virus and side effects of medications. Among many types of cardiac arrhythmia, atrial fibrillation (AF) is the most common arrhythmia associated with a high risk of blood clots in the heart, leading to a life-threatening condition such as stroke [3-5]. Along with the changes in the modern lifestyle and ageing society, the statistic of suffering AF is estimated to increase over the next 30 years. Even in Asia alone, 72 million people are estimated to suffer from AF with 4 percent of them being associated with stroke in 2050 [6].

To reduce the fatality rate of AF, fast and early detection has become highly significant. Conventionally, physicians capture the arrhythmias using 12-lead ECG devices or Holter monitoring devices. However, conventional methods used by physicians to detect AF has the potential to misclassify the beats with human naked eyes only from reading a long ECG record since the nature of AF in ECG cannot be detected from a single beat of ECG. In addition, it takes years of experience or highly trained physicians to interpret and analyze complex waveforms of ECG signals properly. Moreover, ECG signals usually involve noises such as powerline inferences, baseline wander, and electromyographic noises resulting in the misinterpretation of the heartbeat [7]. For these reasons, automatic diagnosis of ECG becomes highly demanded to assist physicians to detect arrhythmias automatically and accurately [8, 9].

Along with the increasing trend in the artificial neural network, a large number of published works have presented classification work by using the deep learning neural network approach. For example, [10-12] applied artificial neural networks to classify atrial fibrillation (AF), ventricular fibrillation (VF), premature atrial contraction (PAC), premature ventricular contraction (PVC), right bundle branch block (RBBB) and left bundle branch block (LBBB) from the normal beat. However, the techniques applied in those are not suitable to be deployed in portable ECG devices due to the need for high computational resources for intensive data processing with high cache memory consumption. The higher accuracy demanded from the network, the more parameters to process is needed in the conventional

neural network model. Increasing the size of the model's parameters results in a higher utilization of computational resources.

To address these issues, researchers proposed the use of lightweight neural networks which can be deployed on mobile phones or limited resources [13-17]. The difference between lightweight neural networks from conventional ones is the approaches with lower numbers of parameters to process resulting in reducing the model size, and memory footprint while preserving the high speed and accuracy in the output result [18].

To implement a neural network-based cardiac arrhythmias classification model with portable devices, a study developed an AF classification system with the concept of using compressive sensed ECG instead of ordinary ECG signal on a lightweight deep learning network MobileNetV2 [19]. Regarding drawbacks, this study only performed the classification of normal and atrial fibrillation, and the accuracy was 87% because of the loss of some information due to compressing the original signal. In a subsequent research [20], a combination of a low complexity algorithm (LDA) in the portable device and AlexNet implemented in the cloud to differentiate AF from normal rhythm was proposed in . This approach required two computational platforms (portable device and cloud) to perform a single classification task and was less cost-effective due to the yearly subscription of the cloud platform. In another study [21], a custom design of a lightweight deep neural network, based on MobileNetV2 to classify PAC, PVC, NSR (normal sinus rhythm), LBBB (left bundle branch beat) and RBBB (right bundle branch beat) was proposed. The proposed system only intended to classify the arrhythmias beat by beat, which makes it unsuitable for the classification of long-term rhythm-based arrhythmias AF.

The purpose of this study is to detect atrial fibrillation (AF) from normal sinus rhythm (NSR) by utilizing a lightweight deep learning neural network that is suitable for deployment in mobile devices with limited computational resources and memory constraints. This study will utilize 30-second ECG interval segments for classification since the ECG record with at least 30-second episode are clinically recognized as AF [22]. The 30-second ECG signal will then be transformed into an image. The 30-second ECG signal will then be transformed into an image, which will serve as input for the neural network to classify cardiac arrhythmias. In addition to previous works, this study will also include the classification of premature atrial and ventricular contractions (PAC and PVC) occurrence was developed within normal sinus rhythm 30-second interval , as these conditions can lead to AF gradually [23-25]. Moreover, PAC is often misdiagnosed as a normal heartbeat and is referred to as the "Silent Killer," making its detection critical [25]. Therefore, this study aims to classify dual nature of cardiac arrhythmias in one system.

This study is particularly beneficial for those living in rural areas, where access to cardiology specialists and equipment for monitoring arrhythmias is limited. Additionally, the combination of deep learning and portable ECG access eliminates the need for trained personnel to detect abnormal heart rhythms in patients, including those with COVID-19 who require constant vital health monitoring. Early detection of arrhythmias can improve treatment outcomes, and this study has the potential to enhance treatment options for doctors.

Overall, this study provides a novel approach to detecting and classifying arrhythmias using a lightweight deep learning neural network suitable for mobile devices. Its impact can be far-reaching, particularly for underserved communities, where access to advanced healthcare is limited. Therefore, this study will have significant implications for improving patient outcomes and advancing medical research.

1.2 Objective

1. To classify the targeted cardiac arrhythmias (NSR, AF, PAC, and PVC) with the up-to-date lightweight deep learning neural network model
2. To investigate the appropriate input parameters insisting on the classification performance of the system
3. To investigate and demonstrate the deployment of trained deep learning neural networks on mobile platforms

1.3 Scope of Thesis

The scope of this thesis is as follows:

1. To design a system capable of classifying two types of cardiac rhythms, namely atrial fibrillation, which requires series of beats for detection, and premature atrial and ventricular contractions, which can be detected on a beat-by-beat basis
2. To ensure consistency of ECG records used in this study by limiting them to 30-second duration segments and exclusively labeled for specific arrhythmias excluding any other ones
3. To develop a compact image transformation approach that can capture the salient features of the targeted arrhythmia, thus enabling efficient classification
4. To explore and select the state-of-the-art lightweight deep learning model such as MobileNetV2, and EfficientNet for arrhythmia classification based on image data

2 Background section

2.1 History of ECG

From 1900 to 03, Dutch physiologist Willem Einthoven was credited for the invention of the practical electrocardiogram. The electrocardiogram (ECG) is a non-invasive diagnosis tool to capture the electrochemical activity within the heart by placing electrodes on the human skin in specific areas. He received the Nobel Prize in Physiology and Medicine for it in 1924 [26, 27]. Because of his invention, the root of cardiac diseases and arrhythmias become interpretable for physicians in cardiology. The invention and applying the electrocardiogram as a clinical application was the upturning point for the development of the science of diagnosis in cardiology. The first commercial model of ECG was sold to Sir Edward Schafer of the University of Edinburgh to use clinically as a medical diagnosis tool in 1908.

The very first electrocardiogram made by Willem Einthoven was based on the string galvanometer. The electrocardiogram has gradually evolved into a common 12-lead ECG. In fact, the physical electrodes to collect the heart electroactivities are placed with 10 electrodes: four on the limbs(the arms and the legs) and six on the chest. The right leg is used as the ground. The three electrodes on the left and right arms and left leg form a triangle, which gives six views : 3 bipolar views (Lead I, II, III) and 3 unipolar views (augmented voltage right (aVR), augmented voltage left (aVL), and augmented voltage foot (aVF)). The six electrodes placed on the chest give the remaining information of 6 leads (views). In this way, the physicians can collect anatomical information about the heart happening within the patient as shown in Fig 2.1.

However, the 12-lead ECG is not flexible and unable to monitor the heart during daily activities. In this case for out-patients, Holter monitoring devices, named after the physicist Norman Holter, are used to monitor the heart activity inactive not just at rest to capture the arrhythmias over 24-48 hours. The Holter monitoring device as shown in Fig 2.2 has three to eight electrodes on the chest connected to a small device that is attached to the patient's waist. Gradually, single-lead portable handle ECG devices such as AliveCor and Omron have become trendy along with the development in mobile health care and sensor technologies. Moreover, ECG is now included in the smartwatch-like Apple watch which can be applied to in-home care patients and athletes [28].

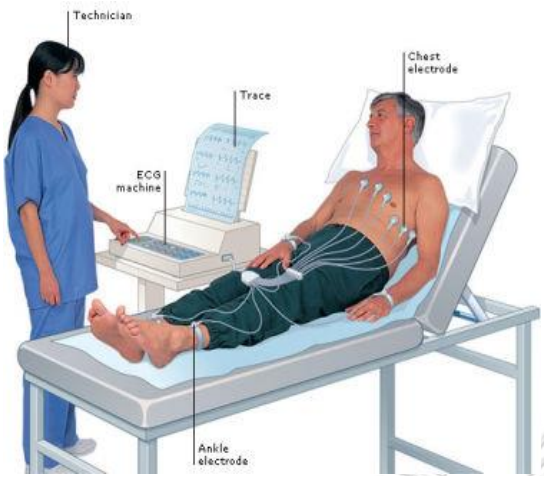


Fig 2.1. The common clinical 12-Lead ECG [29]

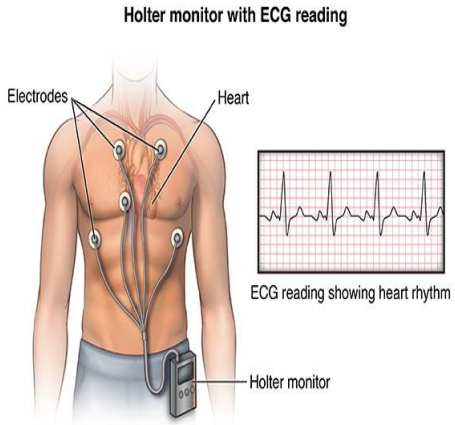


Fig 2.2. The Holter ECG Monitoring Device [29]

2.2 Interpretation of electrocardiograph

The electrocardiogram (ECG) is a vital diagnostic modality that enables visualization of the electrical activity of the heart, facilitating assessment of its functional status. This diagnostic technique involves the use of an electrocardiograph to capture the electrical signals generated by the cardiac system. To fully comprehend the information conveyed by an ECG, it is essential to have a solid understanding of the fundamental principles underlying the heart's electrical system, including the representation of the action potential vectors on the ECG.

The heart conduction system controls the generation and propagation of electrical signals or action potentials triggered at the sinoatrial (SA) node, the heart’s natural pacemaker in the right atrium. Consequently, it stimulates the heart muscles to contract and the heart to pump oxygenated blood. For the heart muscle cell to contract, it must depolarize, that is, the ion balance on either side of its cell membrane must change suddenly and in such a way that the inside of the cell becomes less negatively charged. All electrical cardiac impulses should not only take from the SA node but should propagate a normal conduction pathway, which is originated from the SA node, passed through the atria and then the AV node, through a bundle of HIS and its branches, and Purkinje Fibers as shown in Fig 2.3.

This electrical activity can be measured by electrodes placed at specific points on the body skin from which a composite recording is produced in the form of a graph. This recording is known as an electrocardiogram or ECG, sometimes referred to as EKG. The electrocardiogram or ECG represents the overall trace of electrochemical activity widely

spreading from many cardiac action potentials. ECG can be interpreted by observing its significant wave construction: P, Q, R, S and T.

The signal spreads across both atria causing the cardiac muscle cell to depolarize and contract to induce a phase known as atrial systole. This activity appears as a P wave on the electrocardiograph. After leaving the atria, the signal stimulates the ventricles via the atrioventricular or AV node located in the interatrial septum for the next procedure. It then enters the bundle of HIS and spreads through the bundle branches and the large-diameter Purkinje fibers along the ventricle walls. As the signal spreads through the ventricles, the contractile fibres depolarize and contract very rapidly inducing ventricular systole. This rapid ventricular depolarization can be regarded as a QRS complex form in the electrocardiogram. Atrial repolarization also occurs at this, but any atrial activity is hidden on the ECG by the QRS complex. In the end, the cardiac tissues in the ventricles start to reacquire the relaxed state after the signal passes out of the ventricles, described as ventricular diastole. The rounded curving after the QRS complex on the ECG is a T wave representing ventricular repolarization at which the ventricles start to relax. The duration between ventricular repolarization and depolarization can be known via the ST segment. The sequence of events just described and the associated trace repeats with every heartbeat. ECG is not a tracing of a single action potential but a combination of the many action potentials that constitute the electrical activity of the heart [29, 30].

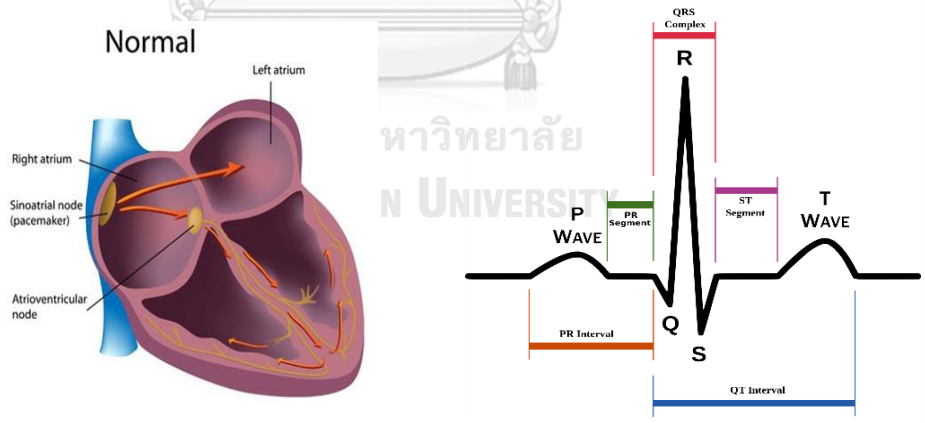


Fig 2.3. (left) Cross-sectional view of the flow of the heart's electrical system [50]
(right) Normal ECG diagram for a human heartbeat as seen on ECG

2.2.1 Heart Rate, Rhythm, and Isoelectric baseline

When analyzing an electrocardiogram, clinicians typically assess the rhythm and heart rate to determine whether the patient is experiencing arrhythmias. Heart rate is typically considered normal if it falls within the range of 60 to 100 beats per minute for an average adult. To calculate heart rate, one can count the number of QRS complexes (the large spikes on the ECG trace) that occur within a one-minute interval. Rhythm refers to the pattern of the ECG signal, which can be analyzed by comparing the intervals between successive P or R waves. An isoelectric baseline, which represents the resting state of the heart without any electrical activity, can be found as a straight line between the T and P segments on the ECG trace.

2.3 Current Portable Personal Monitoring ECG Devices

From the domestic and personal health care point of view, ECG devices with less than or equal to three electrodes will be regarded as portable personal monitoring ECG devices [31]. Recent studies on the impact of personal monitoring ECG devices for detecting Atrial Fibrillation (AF) have shown that devices such as AliveCor Kardia, Omron HeartScan, and Zenicor ECG, which are shown in Fig 2.4, demonstrate high levels of AF detection comparable to conventional ECG devices.

According to the academic studies [28] analyzing the performance of these personal ECG devices, the AliveCor Kardia device has shown sensitivity values ranging from 71.4% to 98% in correctly detecting positive cases, with specificity ranging from 91.4% to 97% in identifying negative cases. The utility of the Omron HeartScan in detecting atrial fibrillation has been demonstrated in five studies, with sensitivity ranging from 30% to 100% and specificity ranging from 76.2% to 97%. The Zenicor ECG has shown high sensitivity (96%) and specificity (92%) for atrial fibrillation detection. However, false positive diagnoses of AF can lead to unnecessary treatment and burden on clinicians. Further research and development may be necessary to reduce false positives and enhance the overall utility of personal ECG devices in detecting AF and other cardiac conditions.

(a)AliveCor Kardia Mobile Single-lead handheld ECG [28]

(b)Omron Health Scan [28]

(c)Zenicor ECG [28]

Fig 2.4. Commercially available portable personal monitoring ECG devices

2.4 Cardiac Arrhythmia

Cardiac arrhythmia refers to any deviation from the normal rhythm of the heart, which can occur due to abnormal triggered action potential locations within the heart. There are three primary mechanisms responsible for cardiac arrhythmias: automaticity, triggered activity, and reentry.

Automaticity refers to the natural spontaneous action potential generated by cardiac tissue, typically from the sinoatrial (SA) node. If the SA node fails to generate impulses, the automatic impulses can originate from other areas of the heart, such as the atrial tissue.

Triggered activity occurs when premature activation of cardiac muscle cells leads to depolarization and the initiation of impulses. The most common cardiac arrhythmias caused by triggered activity is premature ventricular contraction (PVC) when the ventricles are out of sync with the upper chamber atria.

Reentry causes the heart to beat faster than normal due to the inability of the electrical impulse from the SA node to complete a cardiac cycle, resulting in self-sustaining loop of impulse signals. A prime example of cardiac arrhythmias caused by reentry is atrial fibrillation (AF).

In medical terms, there are two genres of arrhythmia types based on rhythm: tachycardia, which refers to a faster heart rate than normal (>100 bpm), and bradycardia, which refers to a slower heart rate than normal (<60 bpm). Among the many types of arrhythmias, this research focuses on the three most common: atrial fibrillation, premature atrial contractions, and premature ventricular contractions. However, it should be noted that current diagnostic tools, such as single-lead electrocardiograms and modified blood pressure monitors, may have limitations in identifying arrhythmias in primary care settings and could lead to unnecessary treatment [32-34].

2.4.1 Atrial Fibrillation (AF)

Atrial fibrillation is the most found arrhythmia and a serious type of cardiac arrhythmia associated with having a blood clot in the heart leading to consequential health complications, especially stroke. In AF, the heartbeat is irregularly irregular rhythm without P-waves and the absence of an isoelectric baseline in the ECG morphology as prescribed in Fig 2.5. Moreover, F-waves (fibrillatory waves) appear with fine amplitude (less than 0.05 mV) or coarse amplitude (greater than 0.05 mV) that are usually misdiagnosed as P waves utmost 0.25 mV). For this reason, it is better to detect long rhythm appraisal screening for effective AF detection than single rhythm appraisal [35].

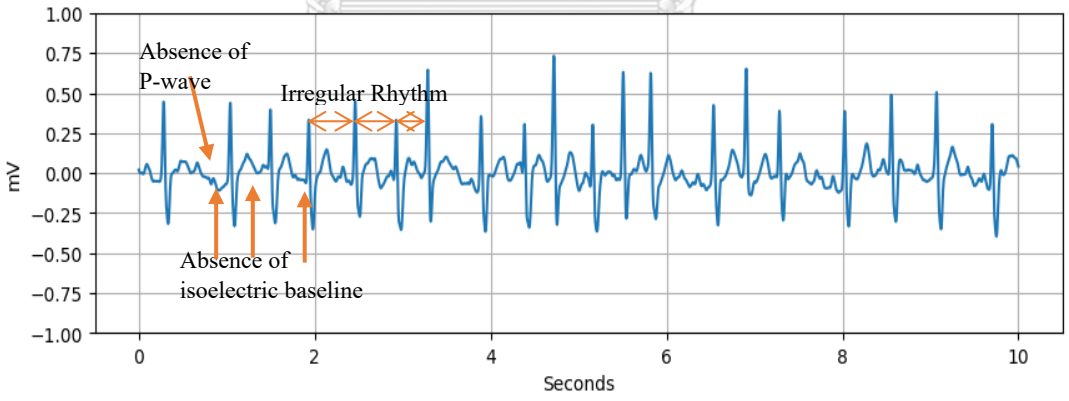


Fig 2.5. 10-seconds of AF ECG signal from Record No: 100

2.4.2 Premature Atrial Contractions /Supraventricular Premature Complexes (PAC)

PAC is the appearance of a beat earlier than it should appear because of the triggered signal within the atrium, not from an ordinary SA node. [23, 25]strongly stated that PAC is the leading association for AF and other serious cardiac-related risks. PAC is unnoticeable to diagnosis because it is hiding in the normal sinus beat. If PAC is detected earlier correctly, the increase of incident arrhythmias can be reduced. For a PAC beat structure stand-alone, its structure is like a normal beat with a non-sinus P wave followed by a normal QRS complex underlying the normal sinus rhythm as shown in Fig 2.6. The natural pacemaker (SA node) is reset from the PAC beat depolarization making the next normal beat delayed. The interval between before the PAC beat and after the PAC beat is less than two times the RR interval, called incomplete compensatory pause.

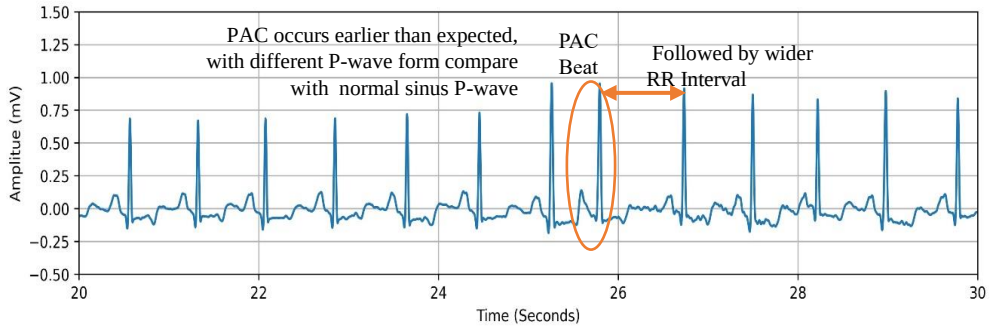


Fig 2.6. The pattern of Premature Atrial Complex within the normal sinus rhythm from Record No: 100

2.4.3 Premature Ventricular Contractions (PVC)

PVC is the occurrence of the ectopic beat within the ventricles leading to interventricular conduction delay. The abnormal morphology for PVC includes the prolonged QRS complex wider than 120 milliseconds and earlier occurrence than expected for the next sinus impulse. Moreover, the discordant ST-ST segment changes in the T wave can be found. Additionally, the phenomenon of backward conduction from the AV node to the atria can occur unintentionally, the PVC complex is followed by a full compensatory pause. The full compensatory pause is the interval between the normal beat before and after the premature beat (PVC beat) equivalent to the length of two normal beats (2 R-R intervals) when added together unlike PAC as below in Fig 2.7.

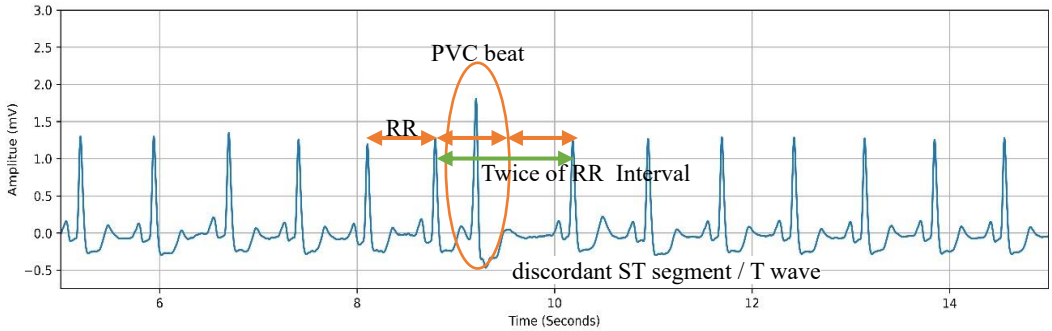


Fig 2.7. The pattern of Premature Ventricular Complex within the normal sinus rhythm from Record No: 105

2.5 Neural Networks

Deep learning is the type of machine learning inspired by the structure of the human brain to improve analytical skills on tons of data-driven processes. The architecture of the neural network is composed of layers of neurons that intimate the function of the human brain. Neurons in the neural networks take in data, train themselves to recognize the patterns in the data and then predict the output for a new set of similar data. The first layer of the network is the input neurons layer that receives the input data, and the last layer is the output layer that predicts the final output. Between the input and output layers, the hidden layers exist where most of the computations required by the network are performed. The features of the inputs are selected by the neurons in the hidden layers without human intervention.

Neural networks can be in various shapes and structures, nevertheless, the backbone is as in the following Fig 2.8. In a neural network architecture, the neurons of one layer have connected to the neurons of another layer through channels. Each of these channels between each layer is assigned numerical values known as weight (w). The input data into the first neuron layers are processed and transformed. After that, the inputs are multiplied to the corresponding weight and then their sum is further sent as input to the neurons in the hidden layers. Each of these neurons is assigned a numerical value called the bias (b) which is then added to the input sum. Furthermore, the values have proceeded through the activation function.

The activation function is a type of transformation function which can learn to decide whether the neuron will be operated or not. The data is then transmitted via the operated neuron to the neuron in the next layer through the channels. In this manner, the data are carried and processed to the output layer called forward propagation. In the output layer, the

neuron with the highest probability value sets off and determines the output value. To optimize the output value, is needed to do backpropagation for error checking using the loss function. From the value of the optimizer, the network keeps updating the weight and learning the features and patterns of the data. Deep learning neural networks are characterized by their depth, referring to the number of hidden layers they possess, which distinguishes them from other neural networks.

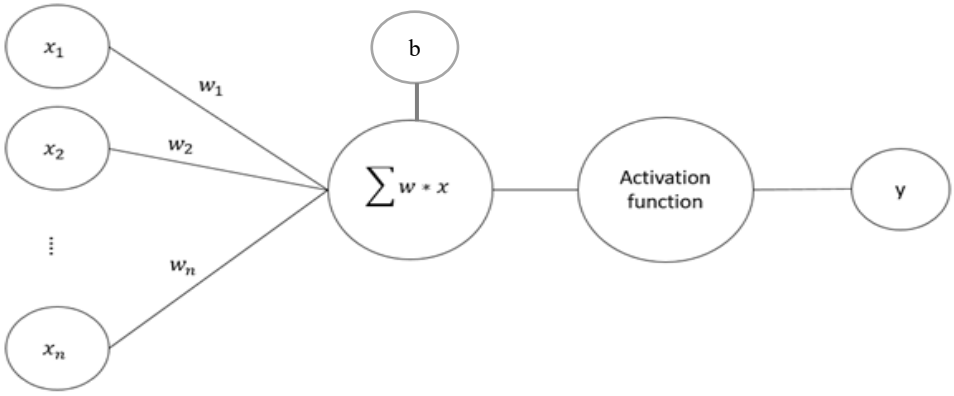


Fig 2.8. Basic Architecture of a Neural Network

2.5.1 Convolutional Neural Network (CNN)

In the last decade, researchers have proposed various learning approaches to neural network architecture, with CNN contributing more than half of researchers' interest in classifying ECG arrhythmias, as stated in [36, 37]. This is primarily due to CNN's effectiveness in classifying problems involving image data as input. The key advantage of applying CNN in image classification lies in its capability to extract significant features by reducing image noise through filtering. Consequently, the model can make more accurate predictions by focusing solely on the essential features of the image.

Firstly, CNN performs convolution to the input image data with convolution matrix as shown in Fig 2.9. The convolution matrix is a matrix of weights which are multiplied with the input image data to extract relevant features. When the convolution filter is applied to the image, it brings out certain features, such as vertical or horizontal lines to make the image more interpretable for the model. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully connected layer is the final layer. The progressive augmentation of depth in a convolutional neural network (CNN) result in heightened complexity, thereby facilitating the discernment of larger segments within the

input image. The initial layers of the network predominantly focus on rudimentary features such as colors and edges. As the image data traverses subsequent layers of the CNN, it progressively acquires the capacity to perceive more substantial components or contours of the object, ultimately culminating in the successful identification of the target object. In general, the more convolution layers the model uses, the more details in the image the model can capture. Nonetheless, a trade-off exists concerning training time as the network's density increases, leading to an enhancement in accuracy. This leads to consuming processing power and memory storage so much that a resource constraint environment is not applicable (such as mobile phones, and embedded system devices) [18].

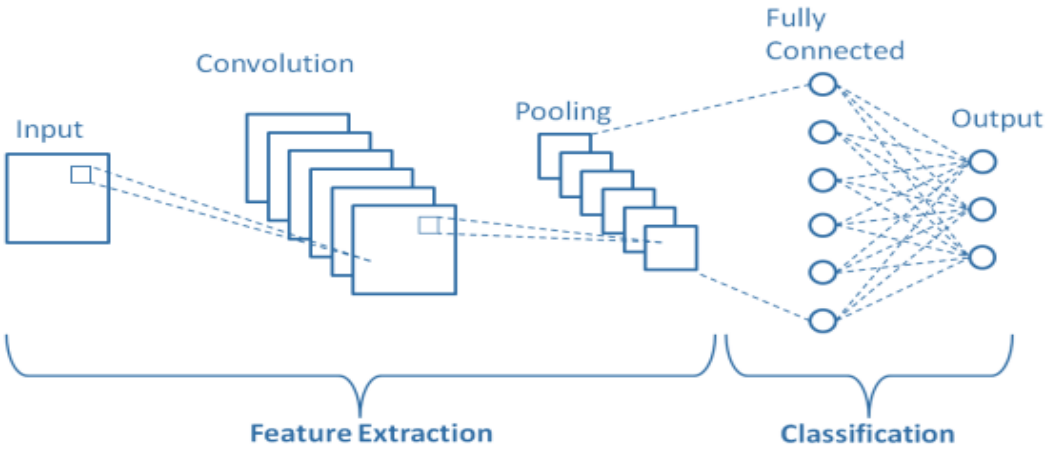


Fig 2.9. Basic Architecture of Convolutional Neural Network

2.5.2 Lightweight Convolutional Neural Network

In recent years, there has been growing interest in addressing the gap within Convolutional Neural Networks (CNNs) through the concept of lightweight convolutional neural networks. These neural networks are a subset of deep learning neural networks that aim to achieve comparable accuracy while significantly reducing the parameter requirements. Parameters in this context refer to the weights of connections that are learned during the training phase or the coefficients of the model selected by the model itself. These coefficients are automatically optimized by the model using an optimization function, which ultimately yields a set of parameters that minimize the error during the learning process. By adopting lightweight convolutional neural networks, researchers strive to strike a balance between computational efficiency and maintaining high levels of performance.

To get the alignment between parameter reduction and accuracy maintenance, new convolution methods have appeared such as dilated convolution, deformable convolution,

group convolution, and depthwise separable convolution. Dilated convolution is the method of using dilated convolution filters which have wide kernel sizes by spacing value kernels with holes presented in Fig 2.10. Dilated convolution filters can improve the learning capacity and enhance the capability of feature extraction by the network without introducing additional parameters to the network [38, 39]. For deformable convolution as in Fig 2.11, it has the adaptability to reshape the convolution kernel by the insides of interest. This leads to being better at extracting necessary features only while increasing the network performance with fewer parameters [40].

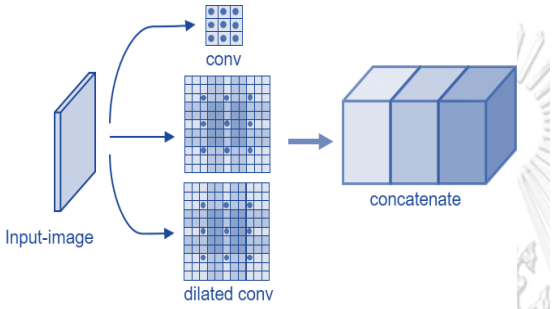


Fig 2.10. Dilated Convolution [38,39]

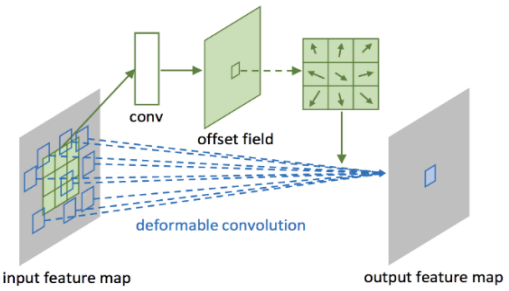


Fig 2.11. 3x3 Deformable Convolution [40]

In group convolution shown in Fig 2.12, the input channel is divided into groups and convolved separately and then joined in each group to give the output result . For the concept of depthwise separable convolution as in Fig 2.13, it is composed of two convolutions; pointwise convolution: 1 x 1 convolution kernel and depthwise convolution: a lightweight convolution filter per input channel. In this process, the computational cost per convolution is reduced since the depthwise convolution performed less than twice of computation compared to conventional convolution neural network. Under these circumstances, not only the size (parameters) of the model but also the consumption of resources (storage and computing power) by the model can be reduced significantly.

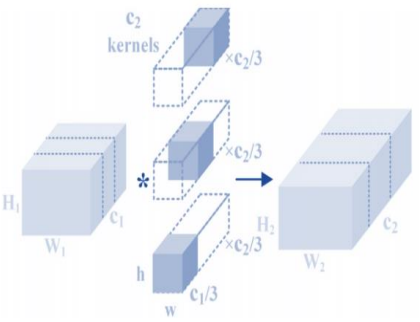


Fig 2.12. Group Convolution [41]

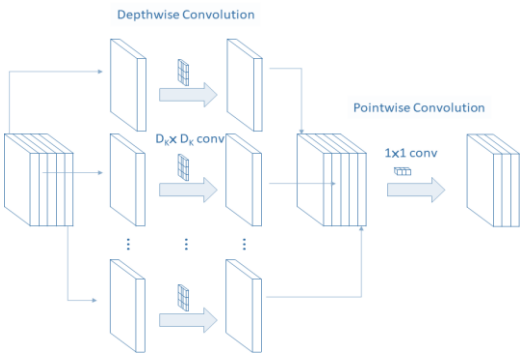


Fig 2.13. Depthwise Separable Convolution [13]

As a reference for judgement, four state-of-the-art lightweight deep neural networks are used as baselines for this research, namely, MobileNetV2, ShuffleNetV1, ShuffleNetV2, MixNet, and EfficientNet. According to the architecture of the models, MobileNetV2 is built with depthwise separable convolution combined inverted residual structure and linear bottleneck, ShuffleNetV1 is built with group convolution through shuffle operation, ShuffleNetV2 is an extension of ShuffleNetV1 with channel shuffling and pointwise group convolution, MixNet is built with by mixing different shades of convolution kernels and EfficientNet is built with convolution included depth, width, and resolution balancing module [13-17, 41].

According to the statistics presented in [18], when analyzing the ImageNet dataset, deep lightweight neural networks such as MobileNetV2, MobileNetV3, and EfficientNet outperform traditional convolutional neural networks by reducing the parameter requirements for calculations. The comparison of parameter reduction between traditional CNN and lightweight CNN, as tested on the Imagenet 2012 Dataset, is shown in Table 2.1.

Table 2.1 Parameters Usage Comparison between Traditional CNN and Lightweight CNN which is tested on 2012 Imagenet Dataset

No.	Traditional CNN	Parameters	No.	Lightweight CNN	Parameters
1	Alexnet	60.9 M	1	MobileNetV2	3.4 M
2	VGG16	138 M	2	MobileNetV3	2.5 M
3	GoogLeNet	6.8 M	3	Efficient Net	5.3 M

MobileNetV2 is a lightweight convolutional neural network developed by Google in 2019. MobileNetV2 is constructed with a full convolutional layer followed by a chain of 17 efficient convolutional building blocks called Inverted Residuals and Linear Bottlenecks (MBConv Block) [14] as shown in Fig 2.14. In this block, the features from a lower dimensional representation are scaled up with 1x1 pointwise convolution with the Relu6 activation function. Then, a Depthwise convolution and Relu6 are applied. ReLU6 function is a variant of ReLU which will be stated in section 2.5.4. ReLU6 bounds the positive output values at 6. Finally, the features are compressed back into the earlier lower dimensional representation with 1x1 pointwise convolution. Since it is used residual connection, it creates a shortcut between the first and last layer for making flow of data in the block easier. By constructing the convolution blocks this way, the trainable parameters are reduced to be faster computation time without decreasing the accuracy and can be deployed easily on the mobile and embedded edge devices.

EfficientNet, developed by Tan and Le [17] in 2019, is a neural network architecture based on the concept of compound scaling. Compound scaling involves adjusting the depth ($d = \alpha\phi$, the number of layers in the network), width ($w = \beta\phi$, the number of channels in the layers), and resolution ($r = \gamma\phi$, the input image size) of the network uniformly, allowing for an extended depth of the network with a compound coefficient, ϕ. EfficientNet uses the MobileNetV2 Convolutional Building Block (MBConv) as the base network by adding the squeeze-and-excitation module (SE). The squeeze-and-excitation module is customizable to change the weighting of the channels by plugging the feature images to a single numeric value using an average pooling. Then, the baseline network can be scaled to get better accuracy and efficiency from EfficientNet-B0 to B7 in accordance with the required system. EfficientNet offers models from B0 to B7, with different scales and input shape requirements. For example, B0 expects input shapes of 224x224, while B1, B2, and others have different sizes. In the research context, a resolution of 224x224 is chosen to compare with MobileNetV2, making EfficientNet B0 the suitable option.

MobileNetV3 is a lightweight convolutional neural network designed for deploying neural networks on mobile phone CPUs, following the success of MobileNetV2 and EfficientNet. In MobileNetV3, a combination of hard swish activation and squeeze-and-excitation modules is applied to the MBConv blocks, as shown in Fig 2.15. Hard swish is an activation function that replaces sigmoid with ReLU6, offering robustness and efficient computation for low-precision settings [13]. A comparison of these three lightweight neural networks (MobileNetV2, EfficientNet, and MobileNetV3) is presented in Table 2.2.

Table 2.2 Composition Comparison for Lightweight Deep Learning Neural Networks interested in this Research

Network	Building Blocks	No of Layers	No of Parameters (millions)	Input Size
MobileNetV2	Inverted Residual Blocks	53	3.4	224 x 224
MobileNetV3	Inverted Residual Blocks Squeeze-and-Excitation Modules	23	2.5	224 x 224
EfficientNetB0	Inverted Residual Blocks Adjustable Squeeze-and-Excitation Modules	66	5.3	224 x 224

2.5.3 Optimizer

Many researchers developed optimization algorithms for a wide range of neural networks to train with. Among them, Adam stood out and showed promising results. Adam stands for Adaptive Moment Estimations which is an optimization algorithm with adaptive learning rate by putting momentum and RMSProp together. Adam is one of the most common and best optimizers that optimizes the neural network to train in less time effectively [42]. The weight updates in neural networks with Adam are performed as per follow equation:

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

In which:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Where Η stands for learning rate, w is the weight , ϵ is a small positive number with $1e^{-8}$ preventing from dividing by zero, g is gradient, t is time, m is aggregate of gradients, ν is sum of square of past gradients and beta parameters β_1 and β_2 present moving average parameter with constant value 0.9.

2.5.4 Activation Function

Activation functions are crucial components of neural networks that introduce non-linearity and determine whether a neuron can contribute to the next layer. They are based on a specific value or threshold, and various activation functions have been created to address this decision. The step function is a simple and intuitive function that activates a neuron if its value is above a certain threshold and does not activate it otherwise. However, it has limitations when classifying multiple neurons into different classes due to its linearity, making it challenging to determine which class a particular neuron belongs to.

To overcome this issue of step function, activation functions that output a range of probabilities were developed. Linear functions can be connected to multiple neurons, allowing for a maximum value to be selected in case more than one neuron fires. However,

the gradient has no relationship with x, making it unsuitable for gradient descent. The sigmoid function is a commonly used activation function in neural networks that has a smooth gradient, outputs analog activation, which is defined as an if x is equal to 1 over 1 plus e to the negative x. This function has a non-linear nature, non-binary activations, smooth gradients, and is bound in a range from zero to one inclusive. Despite these advantages, the sigmoid function has several disadvantages, such as the gradient at the ends of the function is almost zero, giving rise to the vanishing gradient problem, where the gradient becomes too small for gradient descent.

Another commonly used activation function is the tanh function, which is similar to the sigmoid function but is bound in the range of -1 to 1 inclusive. The rectified linear unit (ReLU) function is another popular activation function that overcomes the vanishing gradient problem of sigmoid and tan h functions and is computationally efficient. However, it has its own limitations, such as the "dying ReLU" problem, where neurons can become inactive and produce zero output, leading to "dead" neurons in the network.

The choice of activation function depends on the specific problem being addressed. The graph of the mentioned activation functions are illustrated in Fig 2.14. While the step function, sigmoid, tan h, and ReLU are commonly used activation functions, the effect of each function on this research is explored to understand the advantages and disadvantages of each function to determine which one is best suited for this research.

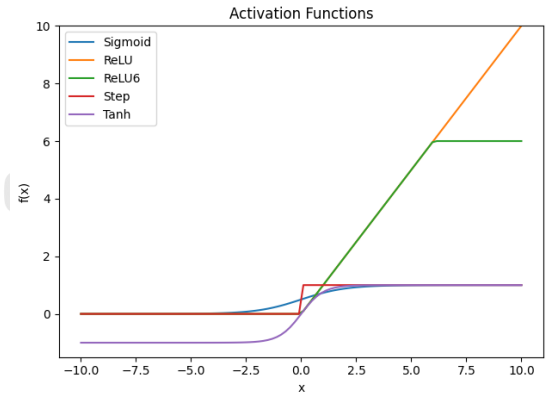


Fig 2.14. Plot of Five Activation Functions

2.5.5 Loss Function

The cross-entropy loss function is widely utilized in machine learning, with binary cross-entropy for binary classification and sparse categorical cross-entropy for multiclass problems. In this particular study, sparse categorical cross-entropy is selected as there are four classes to categorize. This loss function penalizes the model for low probabilities for correct

classes and high ones for incorrect classes. The objective is to reduce the difference between predicted and true class distributions. This can be achieved by updating model parameters via backpropagation. Sparse categorical cross-entropy is more efficient for larger multiclass problems, and regularization can be introduced for handling noisy data. The equation of sparse categorical cross-entropy loss is as follow:

$$\text{Loss} = -\sum_{k=0}^{n} y.\log Y_k$$

where y is true value and Y is predicted probabilities of the classes.

2.5.6 Regularization

Among many forms of regularizations to increase the accuracy of deep learning, the simple and most common one : dropout is applied in this research with different values for different trials to prevent over fitting by temporarily suspended the connections within input and hidden at specific probability.

2.5.7 Normalization

In deep learning neural networks, normalization layer plays big parts to accelerate the model learning rate, optimize the performance and reduce overfitting. In this research , batch normalization is applied in both training and testing. Batch normalization is created by two researchers Sergey Ioffe and Christian Szegedy [43]. Instead of only normalizing the input data and then feeding the data into the neural network, all the outputs of all the hidden layers in our network are normalized. The equation applied batch normalization for a mini batch, BN is as follow:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i = \gamma\hat{x}_i + \beta = \mathrm{BN}_{\gamma,\beta}(x_i)$$

Where γ and β are learnable parameters.

3 Literature Review

Accurate detection and classification of cardiac arrhythmia types are critical for effective treatment, however, there are still challenges for clinicians due to the complex and variable nature of arrhythmias. In recent years, machine learning and artificial intelligence have emerged as promising tools for computer-aided diagnosis of cardiac arrhythmias. Support vector machine (SVM), principal components analysis (PCA), and k-means clustering are among the most widely used machine learning algorithms for arrhythmia detection.

In the literature, four basic steps are generally performed for arrhythmias classification: preprocessing, heartbeat segmentation, feature extraction, and classification. The abnormality of ECG signals classified from the time-series signal requires a combination of multiple steps such as PCA for features dimension reduction and selection, a bag of visual words for clustering, and then SVM and Random Forest as a classifier to achieve high accuracy to develop hybrid cardiac arrhythmia classification model using multiple machine learning in [44]. This proposed system enables classifying 13 classes among 16 arrhythmia classes in Arrhythmias Dataset __UCI Machine Learning Repository with an accuracy of 91.2%. Alternatively, Hsu and Cheng [45] proposed a supervised machine learning model with features extracted from waveform-based signal processing (WIP) to the original signal and a deep learning model with a 2-D image to classify normal beat (N), supraventricular ectopic beat (S), ventricular ectopic (V), fusion (F), and unknown (Q) from the MIT BIH Arrhythmias database. Preprocessing the signals is compulsory to analyze time-series signals due to the existence of noises, such as power line interference, motion artefact, baseline wander, etc. These conventional machine learning approaches required signal preprocessing, waveform detection, feature extraction and the use of manually reconstructed features selection for classification with statistical computing tasks. By synthesizing and critically evaluating the existing literature a comprehensive overview of the state-of-the-art methods for arrhythmia detection using machine learning, highlighting recent advances, challenges, and future directions will be provided sectors by sectors.

3.1 ECG Classification with Time-series Data and Neural Network

Mathews et al. [10] proposed a method for classifying five types of cardiac arrhythmias (Normal beat, Premature Atrial Contraction (PAC), Premature Ventricular Contraction (PVC), Fusion beat, and unknown beats) from the MIT BIH Arrhythmias Database. They employed a deep belief network (DBN) with three layers to create a low-resource consuming system by utilizing a low sampling rate and a small set of features. Prior to implementing the DBN, the authors carried out three preliminary steps.

Firstly, baseline wander, power-line interference, high-frequency noise, and motion artefacts are removed not to disturb analysis performance. Secondly, the preprocessed signals are downsampled from the original 360 Hz to 114 Hz for heartbeat detection. The heartbeat-detected signals are then segmented per beat for T-wave offset detection. Finally, feature extraction proceeded with two practices: the former contained 26 features values of pre-RR intervals, post-RR intervals, average-RR intervals, heartbeat intervals and segmented morphology intervals and the latter holds 22 features from RR intervals between successive heartbeats and fixed-interval morphology.

The proposed method achieved the objective of classification in data with low resolution. According to the confusion matrix, this approach is quite sensitive at distinguishing PAC and PVC with 93.63% and 95.87% respectively although the normal beats are misclassified as PAC. The manual features selection is the major drawback to their experiment leading to that this approach not being appropriate for those who need a medical perspective background and is not reliable for the classification of rhythm base arrhythmias clinically.

Kiranyaz et al. [11] developed a method comparable to Mathews et al. [10]for classifying five cardiac arrhythmias on a patient-specific basis. They utilized an adaptive 5-layer 1-D convolutional neural network (CNN) as both a feature extractor and classifier. The primary aim of this method was to reduce computational complexity, making it suitable for lightweight applications. The study utilized raw data from the MIT BIH Arrhythmias Database and segmented 5-minute data frames from each patient's record for initial patient adaptation training.

The segmented data were fed into the input layer of the proposed adaptive 1-D CNN which included three hidden layers with 32, 16, and 10 neurons respectively utilized 50 iterations of training with backpropagation. The experimental results demonstrated impressive performance for Premature Atrial Contraction (PAC) and Premature Ventricular Contraction (PVC), achieving accuracies of 97.6% and 99% respectively. This study presents a reliable

and innovative approach to address the challenges of training data requirements and computational processing constraints in lightweight applications.

However, a major drawback of implementing this system is its inability to detect arrhythmia characteristics in healthy individuals without any arrhythmias. As mentioned by [46], Kiranyaz's approach heavily relies on the end-user's history to distinguish abnormal beats from normal ones. Additionally, this proposed system is focused on specific beats, limiting its applicability to the most common type of rhythm abnormality, such as Atrial Fibrillation (AF), which requires analysis of a series of beats (rhythm-based).

3.2 ECG Classification with data transformation

Unlike classifying cardiac arrhythmias from the time domain, frequency domain, and considering specific characteristics, patterns, or criteria that are relevant from a medical perspective with complex mathematically and statistically selected features using the time-series signal of the ECG, Salem, et al.[47] proposed the time-series ECG signal to project from 1-D to the 2-D plane by using spectrogram images. As an example, the spectrogram image representing AF and NSR is shown in Fig 3.1 and 3.2. The authors claimed that a large volume of data was required to train a deep neural network for high accuracy whereas the availability of public data in the ECG domain is quite small. Therefore, the authors studied the classification of cardiac arrhythmias using transfer learning from a pre-trained 2-D convolutional neural network to reduce the requirement of high data volume while maintaining high performance.

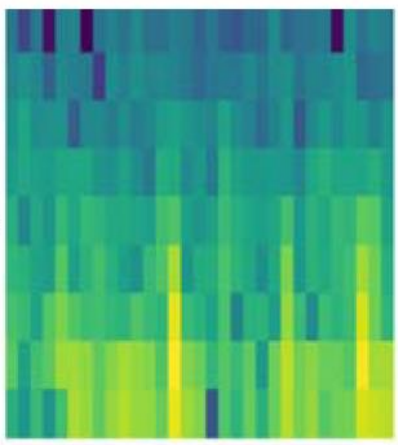


Fig 3.1. Spectrogram Representation of AF from Salem, et al. [47]

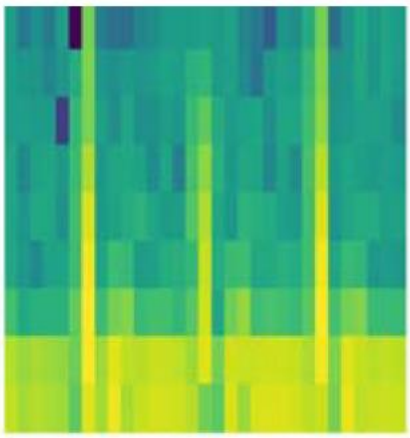


Fig 3.2. Spectrogram Representation of NSR from Salem, et al. [47]

This study analyzed 7008 data instances from four datasets: MIT-BIH Normal Sinus Rhythm Database, MIT-BIH Atrial Fibrillation Dataset, MIT-BIH Malignant Ventricular Arrhythmia Database, and European ST-T (ST segment and T wave changes) Database: to classify normal conditions, atrial fibrillation, ventricular fibrillation, and changes in ST-segment. The raw ECG signal was cropped into a 500-sample window size around the relative annotated area. After that, the signal was transformed into a spectrogram image using the Fourier Transform method. This alternative approach took the transformed spectrogram images as the input of the neural network, which was pertained with millions of images in the ImageNet, for feature extraction. The extracted features were then classified according to the classes with the support vector machine classifier and achieved 97.23% accuracy in classification. The implementation of transfer learning showed that the models could analyze quite correctly the unseen spectrogram images of ECG signals with the knowledge gained from the pre-trained model.

Another similar approach to converting the 1-D signal to the 3-D image is with a recurrent plot to cover not only the spectral features from the signal but also included temporal features from the original signal in the image. A recent study by Mathunjwa, et al. [48] presented an approach to transforming raw ECG data into an image using the recurrence plot (RP) technique. Recurrent plot is a visualization matrix which can visualize the phase space. The property of RP is the capability of presenting the phase nature of the time-series signal. The authors tried to solve the need for arrhythmia classification in devices without relying on the users' medical history. The authors analyzed the data from MIT BIH AF database, MIT BIH VF database and Creighton University Ventricular Tachyarrhythmia Database (CUDB). In this research, the raw ECG signals with 30-second were first segmented into 2-second segments. Then, the segments were selected based on the annotation approach used for the ECG data. After that, selected two-second segments of the 1D ECG signals were converted into 2D images using recurrence plots (RPs). The example representation of the RP image for AF and NSR used by the research is illustrated in Fig 3.3 and 3.4.

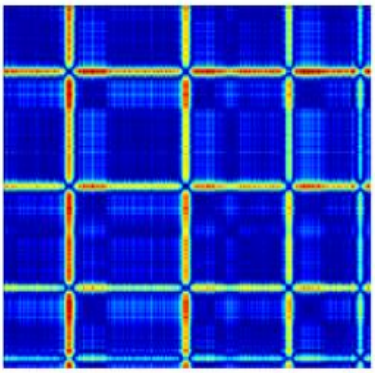


Fig 3.3. Recurrent Plot of AF from Mathunjwa, et al. [48]

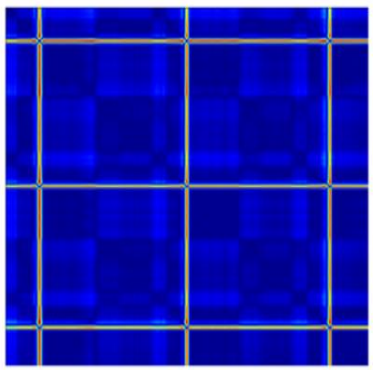


Fig 3.4. Recurrent Plot of NSR from Mathunjwa, et al. [48]

The altered RP images were used as the input to the neural network. This system had performed two classification steps: one with 2-second segment data for classifying noise, ventricular fibrillation, and others and the second step classifies the others signal with beat-by-beat classification to AF, normal, PAC and PVC. The classification was performed directly on neural networks: AlexNet, VGG 16 and VGG 19 and compared the results of each network respectively. To get the optimal values, the authors performed many trials with various learning rates and epochs. Finally, the results showed that AlexNet outperformed in classification compared to VGG 16 and 19 results. The accuracy achievement of AlexNet to classify arrhythmias from RP is 96.64 %.

However, this study has a limitation that hinders implementation in resource constraints environment. This study classified rhythm-based arrhythmia AF to distinguish with beat leading to a false positive result as PAC. The authors themselves stated that this approach requires a lot of processing power and storage which cannot fill the gap of implementation in devices. Moreover, the transformation of RP is deeply reliant on the threshold value which is directly proportional to the noise existence of the signal. The drawback of these transforming methods requires additional processing steps and consumes processing power and energy for resource-limited devices.

3.3 ECG Classification approach through neural networks optimized for efficient deployment on portable devices

In the study, Abdelazez et al. [19] introduced a classification system for atrial fibrillation (AF) that utilized compressive sensed ECG signals instead of traditional ECG signals. The purpose of this approach was to decrease memory consumption in the system. The compressive sensed ECG proposed by the author is that the time-series 30-second ECG signals are compressed with wavelet transformation. The compressed signal is then transformed into spectrogram images. The resulting transformed images are used to train the lightweight deep learning network MobileNetV2 in detection of the ECG signal whether the signal is AF or not. According to statics shown, However, the compression ratio was found to have a negative impact on classification accuracy. The accuracy of the model predicted declined from 87% accuracy (when utilizing uncompressed signals) to 79% accuracy when compressing approximately one-third to one-fourth of the original ECG signal. This drop in accuracy was attributed to the information lost during the time-series signal compression process.

Chanthercrob, et al. [20] proposed a combination of a low complexity algorithm (LDA) in the portable device and implementation of AlexNet in the cloud to differentiate AF from normal rhythm. Firstly, the heart rate variability (HRV) is calculated based on the R-R interval to know whether it is higher than the normal heart rate or not. The LDA implemented in the device classifies whether the signal is AF or not from the heart rate variability. To improve specificity, the ECG record is divided into the 3-s interval and these time series are transformed into 3-second segment ECG image to classify with cloud-computed AlexNet. However, this approach requires two computational platforms (portable device and cloud) to perform a single classification task and is not cost-effective due to the yearly subscription of the cloud platform [31].

Recently, Liu, et al. [21] performed research on the lightweight network to classify arrhythmias by transforming time-series signals to beat images with high accuracy which is intended to deploy with mobile terminals or embedded devices. Like other researchers, the MIT BIH arrhythmias database was used. In this study, the raw ECG signal was first preprocessed with noise-cancelling by wavelet transform. The clean signal was then segmented into a 1-second time frame and then the plotted signal was saved as an ECG beat image. After that, the images were used as the input of the proposed neural network. The uniqueness of this study was performed on a lightweight neural network with small-scale

convolution filters. The neural network in this research used VGGNet architecture as a backbone with modifications such as removing certain convolutional layers and replacing with the inverted residual with linear bottleneck structure from MobileNet V2. The lightweight neural network served as both feature extractor and classifier in this study which removed the additional computing kinds of stuff and advanced feature engineering concept which affected the performance in resource-limited environments. This research achieved an overall accuracy of 99.41 % although there were some drawbacks to be pointed out. According to the confusion matrix presented, PAC and PVC were misclassified as normal beats. Since this classification was based on analyzing individual beats, it could not be as effective in identifying rhythm-based arrhythmias, such as atrial fibrillation (AF), which requires an analysis of the entire rhythm pattern and the relationship between consecutive beats. The authors stated that this study could use time consumption for computing whereas the memory requirement problem had not been solved.

Table 3.1. Summary of comparison of state-of-art arrhythmias classification using neural networks for deploying on portable devices

Approach	Classified Types	Applied Neural Networks	Data Transformation Techniques	Advantages	Disadvantages	Accuracy
Abdelazez et al. [19]	AF , NSR	MobileNetV2	Compressed 30s Time-series, then transform spectrogram	Reduced memory consumption	Classify AF only, Less Accuracy, Signal Loss	87% (AUC)
Chanthercrob et al. [20]	AF , NSR	LDA and AlexNet	Transformed 3s Time-series into RGB Image	Combination of rhythm-based and morphology-based algorithm	Classify AF only, two computational platforms, cost-effectiveness concerns	96.7% (sensitivity) 100% (specificity)
Liu et al. [21]	NSR , PAC , PVC, LBBB , RBBB	Custom DNN(based on VGGNet and MobileNetV2)	Transformed 1s Time-series into ECG beat Image	Less Processing Time	Unapplicable to AF, Require large data set	99.41% (Accuracy)

4 Methods and Materials

4.1 The Proposed Cardiac Arrhythmias Classification System

The proposed research procedure comprised several key steps for investigating and analyzing electrocardiogram (ECG) data for classification of cardiac arrhythmia detection. The overview step of the proposed cardiac arrhythmias classification system is illustrated in Fig 4.1 and details as follow:

1. Preprocess ECG Data: The raw ECG data will be preprocessed such as filtering techniques to remove noise and artifacts for good quality, segmenting the data into individual heartbeats, and transforming the time-series data into image data to be used as the input image of the network.
2. Create Datasets: In order to train, validate, and test the performance of the models, datasets are divided into three subsets: training, validation, and testing. The training subset is to optimize the model's parameters, the validation subset is to fine-tune the model and avoid overfitting, while the testing subset is for an independent evaluation of the model's performance.
3. Build Input Pipeline: An efficient input pipeline before feeding it into the model plays an important factor in training the model smoothly. This will involve resizing the resulted overlapped ECG beats image to a standardized size in accordance with the model requirement, normalizing the data to a common range, and applying augmentation techniques to increase the diversity of the training data.
4. Instantiate Model: A suitable model architecture is set up for arrhythmia detection, based on previous research and literature. The selected model is going to be used as the foundation for subsequent training and evaluation and further deployment in the mobile device.
5. Train the Model: The instantiated model will be trained with the prepared datasets. During the training process, the model will learn to recognize and classify different types of assigned cardiac arrhythmias based on the features extracted.
6. Record and Analyze Performance: The performance of the trained model will be recorded and analyzed. This involved evaluating various performance metrics, such as accuracy, sensitivity, and specificity, to assess the model's ability to correctly classify arrhythmias. The recorded results will be thoroughly analyzed to gain insights into the model's strengths and weaknesses. Based on the findings, the model

is optimized by adjusting its parameters to minimize the training loss and maximize its predictive capabilities.

7. Start Again with Another Network: To explore different approaches and enhance the research findings, the process is going to be repeated with alternative model architectures. This iterative approach allowed for a comprehensive comparison of multiple networks, enabling a deeper understanding of their relative performance.
8. Compare and Conclude Results: The results obtained from different models will be systematically compared and contrasted to derive meaningful conclusions. The strengths, limitations, and trade-offs of each model will be assessed to identify the most effective approach for arrhythmia detection and suggest further research directions.

Overall, this research procedure employed a systematic and iterative approach to investigate the classification of cardiac arrhythmias using the proposed overlapping of the ECG beat images. The emphasis is on data transformation of the raw ECG data, training models with varying architectures, and critically analyzing their performance to gain insights into their effectiveness.

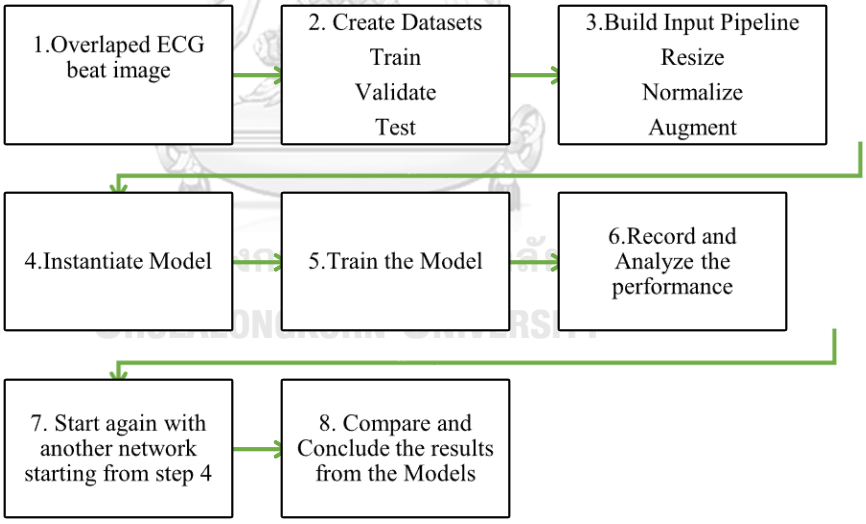


Fig 4.1. Block Diagram of Proposed Research Framework

4.1.1 Database Exploration

The data training for the model in this study will be collected from the clinically certified dataset at PhysioNet namely the Long-Term Atrial Fibrillation (LTAF) database in which 84 records of around 24-hour long ECG recordings with sampling frequency of 128 Hz

including both annotations based on beat and rhythm type in each record [49]. The reason for using ECG data from PhysioNet is due to its provision of annotated electrocardiogram (ECG) data, specifically labeled for beats and rhythms, allowing for the development, and testing of algorithms for beat and rhythm analysis. Furthermore, the user-friendly wfdb Python library offered by PhysioNet simplifies access, analysis, and manipulation of physiological signals. This LTAF database from PhysioNet contains 5 types of beat annotation: N for Normal Sinus Rhythm, A for premature atrial contraction, V for premature ventricular contraction, Q for unknown beat and ‘ “ ’ for missing beat. Comparatively, the rhythm beat annotation has nine types of annotation involved. However, the interest in this thesis is limited to atrial fibrillation and normal sinus rhythm in rhythm-based annotation and normal beat, premature atrial contraction and premature ventricular contraction beat in beat-wise annotation.

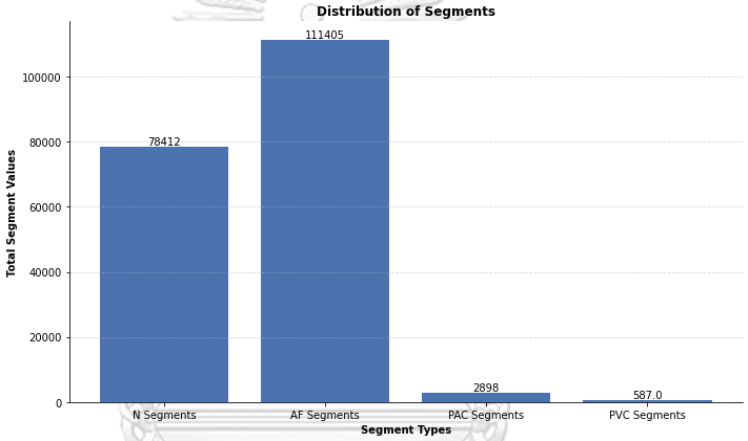


Fig 4.2. The Distribution Chart for 30-second Segments of Each Arrhythmias Type

In Figure 4.2, we illustrate the distribution of resultant 30-second segments after applying our data selection process. The N Segments refer to pure NSR, free from any type of arrhythmias. AF segments represent the number of 30-second segments with AF exclusively. PAC segments consist of PAC beats, with each segment including PAC beats comprising at least 20% of the total beats within that 30-second interval. Similarly, PVC segments consist of PVC beats, with each segment including PVC beats accounting for at least 20% of the total beats within the 30-second period.

To gain deeper insights, Figure 4.3 showcases the distribution of 30-second segments containing premature atrial contractions (PAC) and premature ventricular contractions (PVC), organized according to their respective PAC and PVC beat percentages. The comprehensive details of the methodology employed for selecting 30-second segments from the original database and their subsequent transformation into images are provided in Section 4.1.3.

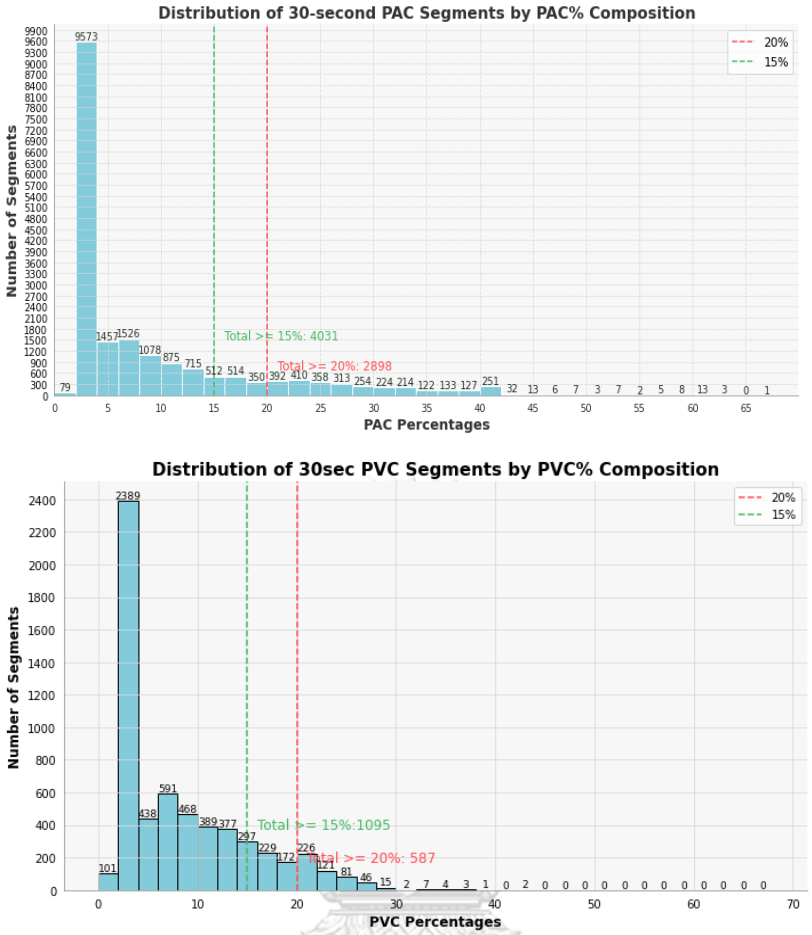


Fig 4.3. Percentage Composition of PVC and PVC Segments

4.1.2 Dataset Preparation

4.1.2.1 Data Selection

Providing the model with relevant data for accurate classification is very critical, particularly in healthcare settings. In this research, the selection of appropriate data is a challenging and time-consuming task. Consequently, this study aims to thoroughly explore and explain the functionality of the algorithms in relation to data selection, offering valuable insights into this intricate process. The dataset utilized for training the neural network encompasses 30-second recordings of four distinct arrhythmia types: atrial fibrillation (AF), characterized by rhythm-type arrhythmias; normal sinus rhythm (NSR); premature atrial contraction (PAC) and premature ventricular contraction (PVC), which represent ectopic-beat type arrhythmias.

The precise assignment of criteria for each arrhythmia is of paramount importance due to the research's dual focus on rhythm-type and ectopic-beat type arrhythmias, which have proven challenging to classify using a single classification model. The original database comprises 84 records, each spanning approximately 24 hours. However, for training the model, we specifically require 30-second data segments. While it may seem that the data is abundant, it is essential to note that each record contains both the desired arrhythmias and those we wish to exclude. Therefore, it necessitates that these two types of arrhythmias do not interfere with each other when we identify episodes of the desired arrhythmias in the data selection. To address this, specific criteria have been devised for each arrhythmia category to identify a suitable 30-second interval from the original electrocardiogram (ECG) recording, as outlined below:

1. Normal Sinus Rhythm (NSR) 30-second record : The whole 30-second duration must be uninterrupted normal sinus rhythm without any additional arrhythmic features encompassing both rhythm-type and ectopic-beat type arrhythmias. We specifically exclude any other types of abnormal heart rhythms or irregular beats from being present in this record.
2. Atrial Fibrillation (AF)30-second record : The whole 30-second duration must be completely atrial fibrillation rhythm without interfered from any other types of irregular beats such as premature atrial contractions (PAC) or premature ventricular contractions (PVC).type of beat arrhythmias .
3. For Premature Atrial Contraction (PAC) and Premature Ventricular Contraction (PVC), specific criteria are established to accurately identify for their analysis. The process involves examining 30-second segments, which must be free from any concurrent rhythm disturbances, such as Atrial Fibrillation (AF) or Ventricular Tachycardia (VT). Furthermore, within these selected segments, a minimum proportion of 20% must be dedicated to either premature atrial or ventricular contractions. For instance, when considering PAC, we carefully check the 30-second segment to ensure that PAC beats make up at least 20% of the total duration, without any PVC beats. Similarly, when studying PVC, the segment should contain at least 20% PVC beats, without any PAC beats.

By adhering to these rigorous selection criteria, the integrity and validity of the resultant dataset are upheld, thereby enhancing the effectiveness of subsequent neural network training processes. The overview representation of the selection is presented as an integral component within diagram 4.4.

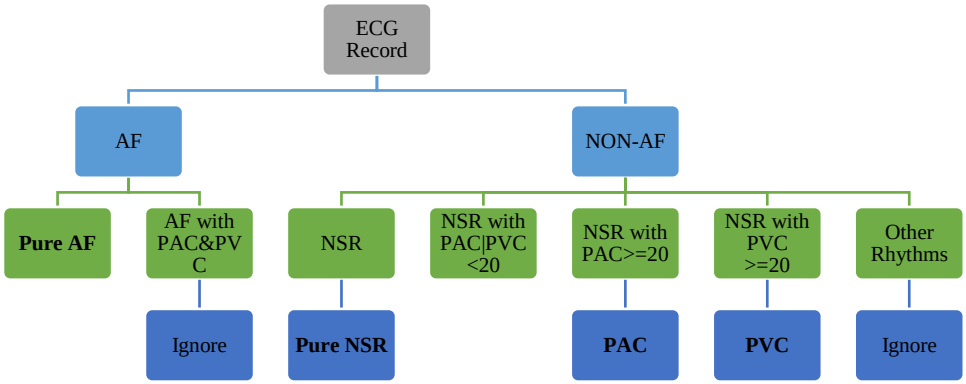


Fig 4.4. Overview of Data Selection and Segmentation Process for Obtaining 30-Second Segments from the LTAF Database

In the process of data retrieval, the Python "wfdb" library plays a crucial role by facilitating access to the original data stored in the PhysioNet database. The database consists of records that are comprised of three distinct file types: "atr," "hea," and "data." To extract the signal values, the ".hea" file type is read. This enables the exploration of various attributes associated with the record, which are stored as a dictionary. In the context of this research, one specific attribute of interest is the "p-signal," which represents a 2D NumPy array containing the physical signal values of the electrocardiogram (ECG) record.

To further gather information regarding the annotation of the record, another file with the extension ".atr" needs to be read. Similar to the ".hea" file, the ".atr" file is also stored as a dictionary, encompassing 14 attributes which are record name, extension, sample, symbol, subtype, channel, aux-note, sampling frequency, label store, description, custom labels, contained labels and annotation length. Among these attributes, the "symbol" attribute is utilized to obtain beat annotations, providing valuable insights into the specific characteristics of each beat. Additionally, the "aux-note" attribute is employed to extract rhythm annotations, shedding light on the overall rhythm patterns present within the ECG recording.

During the data selection process for 30-second segments from the 24-hour record, a window scanning approach is employed to identify the desired signal. The window width is consistently set to 30 seconds, while the step of the window varies depending on the specific arrhythmia type under consideration. For Atrial Fibrillation (AF) and Normal Sinus Rhythm (NSR), the step is also set to 30 seconds. However, for Premature Atrial Contraction (PAC) and Premature Ventricular Contraction (PVC), the step is adjusted to one beat.

In our exploration of the database, as documented in section 4.1.1, it was observed that the data distribution of PAC and PVC is relatively low in comparison to AF and NSR. To address this discrepancy and mitigate potential issues related to data imbalance and

overfitting, a modified approach known as scanning with a window and stepping per beat is employed for data selection. By adopting this method, we aim to compensate for the limited representation of PAC and PVC data while maintaining a balanced dataset. This approach contributes to mitigating the risk of overfitting, ultimately ensuring a more robust and reliable analysis. The algorithm of how the window scanning the record is illustrated in Fig 4.5.

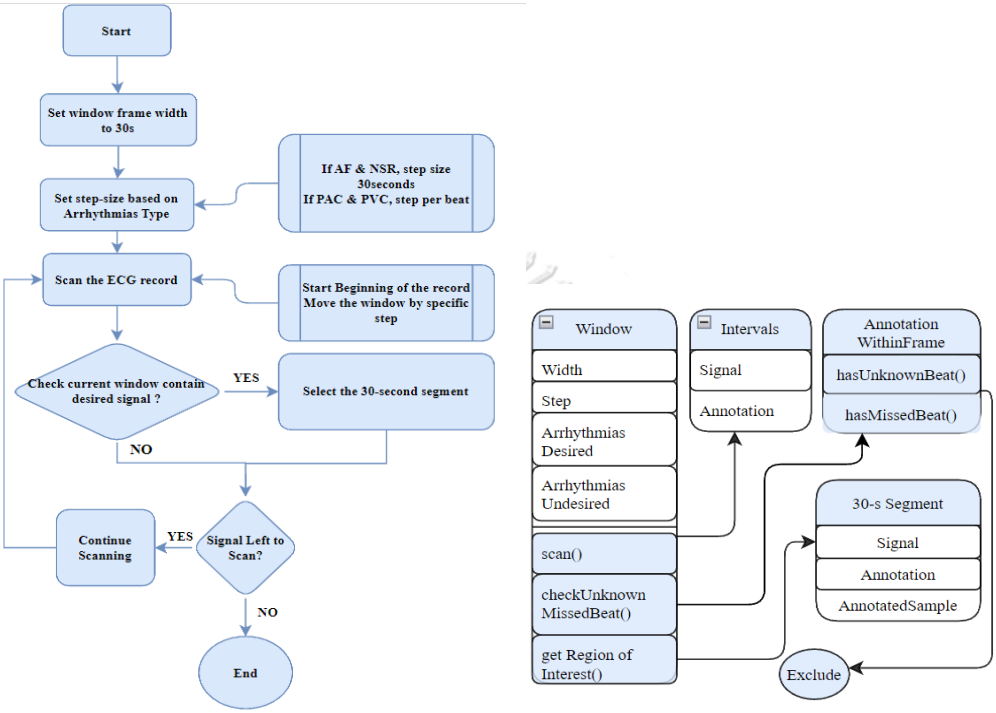


Fig 4.5. Construction of ECG Record Window Scanning :Right Process Flowchart and Left Class Diagram

Initially, the record undergoes scanning to identify intervals exhibiting atrial fibrillation (AF) rhythm, over or equal to a duration of 30 seconds. Subsequently, data within these intervals is collected. Thereafter, the selected intervals in the record are then proceeded into scanning with window to get precisely 30-second intervals segments. The number of samples contained within each 30-second segment is derived by multiplying the sampling frequency, which is 128 Hz for LTAF database, by 30 seconds, resulting in 3840 samples within a single 30-second segment. Furthermore, each selected 30-second segment undergoes thorough examination to identify any presence of beat arrhythmias. Segments that are free from any form of beat arrhythmias are considered as pure atrial fibrillation (AF) segments and are selected for further processing. The same procedure is applied for the interval with only normal sinus rhythm free from any type of arrhythmias. Figures 4.6 and 4.7 present the flow of outlining the step-by-step procedure for extracting AF (Atrial Fibrillation) and NSR (Normal Sinus Rhythm) 30-second segments, respectively, from the original 24-hour long record.

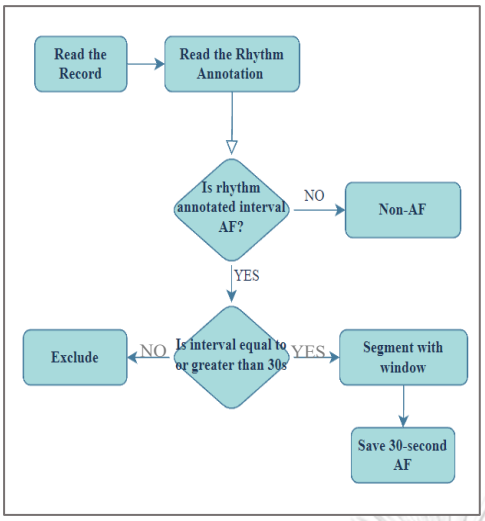


Fig 4.6. Flowchart for Selection of 30-second Interval Atrial Fibrillation Signal

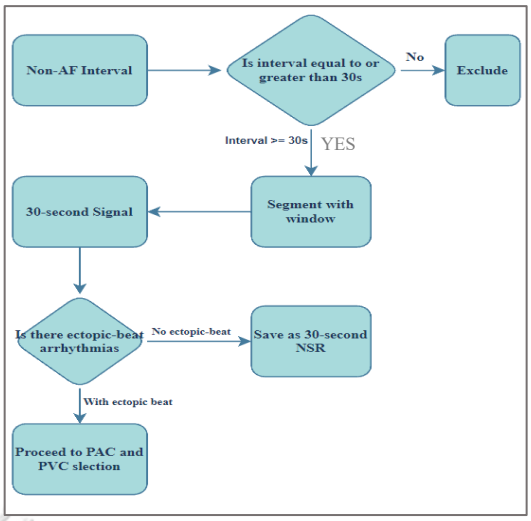


Fig 4.7. Flowchart for Selection of 30-second Interval NSR Signal

In this research, the decision to employ a minimum threshold of 20 percent for the proportion of premature atrial complex (PAC) or premature ventricular complex (PVC) within 30-second segments derives its significance from the insightful findings of Huang et al [50]. By implementing this criterion, our data selection algorithm ensures the preservation of segments exclusively characterized by a PAC and PVC burden of 20 percent or higher, while avoiding any disruption from other forms of arrhythmias, whether ectopic beat or rhythm type. This methodical approach effectively safeguards the integrity of specific segments, facilitating precise analysis and interpretation within the parameters of our study. The process of selecting premature atrial complex (PAC) and premature ventricular complex (PVC) is explained using an outlined step-by-step procedure. This process is illustrated in the flowchart diagram found in Figure 4.8. Additionally, an example is provided to demonstrate how PVC 30-second segments are chosen from the record. Specifically, this example shows what happens when a PAC beat is encountered within the selection window or when the proportion of PVC beats is below the 20 percent threshold. A captured image accompanies Fig 4.9 and Fig 4.10 as the example.

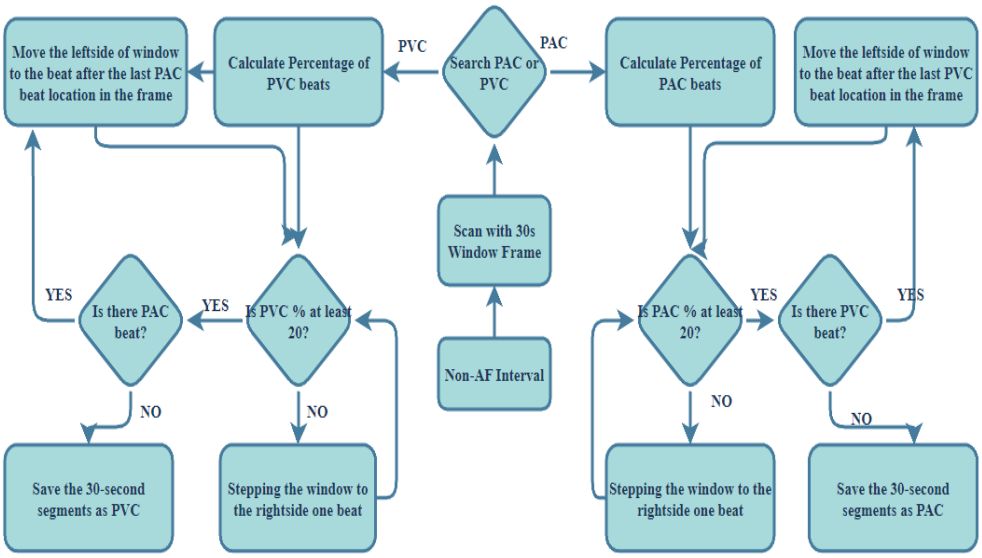


Fig 4.8. Flowchart for Premature Atrial Complex (PAC) and Premature Ventricular Complex (PVC) Selection Procedure

After following the data preparation for each cardiac arrhythmias from the original database, the total of 78412, 11405, 2898 and 587 of 30-second-long ECG signal segments for NSR, AF, PAC (≥ 20%) and PVC (≥20%) respectively. In Table 4.1, a comprehensive summary of key measurements and intervals extracted from the Long-Term Atrial Fibrillation database is provided. The table with each row corresponds to a specific record, identified by the Record ID includes important information such as the number of normal sinus rhythm (NSR) intervals and their total duration, as well as the number and duration of atrial fibrillation (Afib) intervals are presented. Moreover, it provides the presence of premature atrial contractions (PAC) and premature ventricular contractions (PVC) by means of the respective 30-second segment quantities. Furthermore, the table includes columns indicating the presence of PAC or PVC beats with proportions of at least 20% within 30-second intervals.

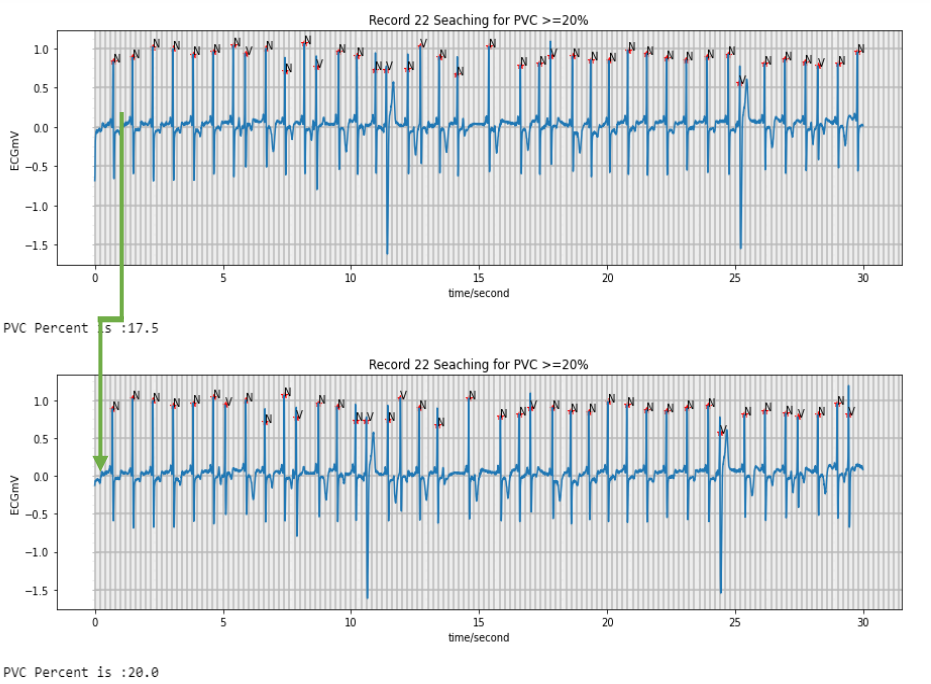


Fig 4.9. The way of sliding window performs while searching for PVC beats contains at least to 20 percent

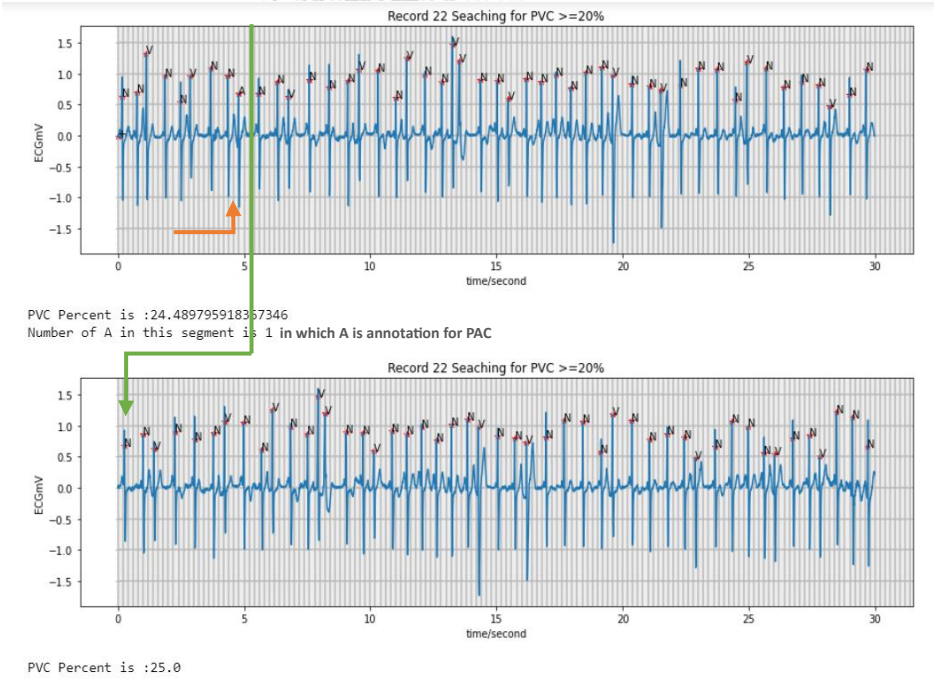


Fig 4.10. Example for showing how the data selecting window works when there is PAC beat (A) while searching for PVC beats at least 20 percent

Table 4.1. Table for Long Term Atrial Fibrillation Database

Rd ID	No of NSR interval	Total Duration of NSR Intervals	30sec Segments (NSR)	No of Afib interval	Total Duration of Afib Intervals	30sec Segments (AF)	30sec Segments (PAC)	PAC % >=20	30sec Segments (PVC)	PVC % >=20
Rd:00	5	18:31:31	2153	44	2:14:54	27	50	0	32	0
Rd:01	457	3:38:53	136	53	16:13:06	1924	202	181	2	1
Rd:03	1648	14:56:49	1379	22	1:20:03	154	454	102	3	0
Rd:05	14	24:17:02	2895	3	0:40:00	0	1099	17	4	0
Rd:06	48	24:22:27	2655	19	0:45:04	45	704	313	0	0
Rd:07	499	21:25:26	1679	8	3:24:47	391	46	0	839	48
Rd:08	20	25:42:48	3073	4	0:01:38	2	340	16	56	0
Rd:10	148	8:23:01	844	80	17:07:44	2001	253	7	9	0
Rd:100	839	21:35:34	1484	659	1:53:30	21	885	161	183	5
Rd:101	675	19:47:36	1538	147	3:20:54	274	563	88	287	24
Rd:102	46	23:07:36	2724	43	51:37:00	80	155	2	24	0
Rd:103	2095	9:37:09	81	1	12:26:55	1492	187	184	0	0
Rd:104	29	17:51:43	2074	16	24:01:00	30	249	10	4	0
Rd:105	398	13:51:55	1357	96	5:11:07	533	57	0	25	0
Rd:11	0	0:00:00	0	2	26:03:09	3083	0	0	0	0
Rd:110	84	22:26:20	2614	22	1:09:57	87	1208	12	0	0
Rd:111	52	15:19:46	1635	26	8:26:24	946	989	12	13	0
Rd:112	1055	13:07:24	867	1042	10:49:36	818	234	2	1	0
Rd:113	19	21:39:33	2506	2	1:58:10	212	52	2	4	0
Rd:114	132	23:35:38	2781	30	12:47:00	22	442	34	4	0
Rd:115	214	3:25:19	201	175	20:15:32	1851	124	7	2	0
Rd:116	752	22:29:29	1885	25	0:24:09	0	1114	120	0	0
Rd:117	4	14:47:19	1771	5	9:11:35	1094	8	0	0	0
Rd:118	7	19:34:24	2156	1	4:11:33	476	392	0	50	0
Rd:119	340	20:50:43	1637	87	2:16:45	0	279	100	9	0
Rd:12	0	0:00:00	0	1	24:05:39	2874	0	0	0	0
Rd:120	53	23:31:17	2370	37	0:17:33	0	746	5	8	0
Rd:121	1335	11:58:04	196	111	3:14:08	277	197	16	230	32
Rd:122	1078	13:39:05	564	16	8:43:01	1024	25	0	949	287
Rd:13	193	10:08:00	1172	2	4:56:14	592	60	1	4	0
Rd:15	4	0:16:00	0	801	23:17:57	1353	0	0	0	0
Rd:16	117	23:14:58	2638	37	0:19:57	10	129	0	1007	63
Rd:17	0	0:00:00	0	1	24:55:10	2942	0	0	0	0
Rd:18	0	0:00:00	0	1	24:59:16	2936	0	0	0	0
Rd:19	83	23:49:19	2824	9	0:05:03	6	1491	20	3	0
Rd:20	0	0:00:00	0	2	24:19:08	2915	0	0	0	0

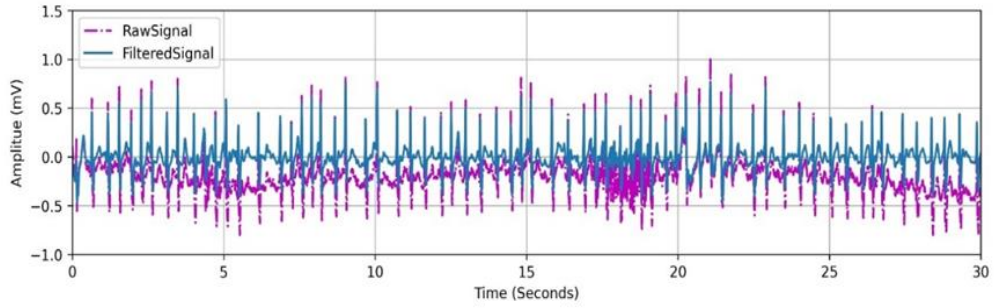
Rd:200	0	0:00:00	0	19	23:50:26	2833	0	0	0	0
Rd:201	0	0:00:00	0	7	23:58:08	2860	0	0	0	0
Rd:202	0	0:00:00	0	1	23:59:51	2807	0	0	0	0
Rd:203	0	0:00:00	0	6	18:09:04	2168	0	0	0	0
Rd:204	0	0:00:00	0	334	20:35:58	1066	0	0	0	0
Rd:205	0	0:00:00	0	1	23:49:35	2814	0	0	0	0
Rd:206	0	0:00:00	0	6	20:47:16	2366	0	0	0	0
Rd:207	0	0:00:00	0	17	23:49:22	2759	0	0	0	0
Rd:208	0	0:00:00	0	1	23:55:16	2861	0	0	0	0
Rd:21	0	0:00:00	0	1	20:56:25	2493	0	0	0	0
Rd:22	1636	22:37:36	382	102	0:23:58	0	576	206	122	51
Rd:23	1194	16:22:18	1130	57	5:59:42	620	774	641	26	0
Rd:24	784	21:58:46	2129	2	0:01:47	2	688	7	8	0
Rd:25	0	0:00:00	0	346	18:01:49	1465	60	0	1	0
Rd:26	93	1:02:02	32	177	16:43:49	990	21	15	22	0
Rd:28	271	15:43:05	1576	106	4:43:36	364	0	0	0	0
Rd:30	14	6:06:41	712	0	0:00:00	0	617	1	9	0
Rd:32	541	19:28:40	1899	361	0:36:26	0	272	22	90	0
Rd:33	0	0:00:00	0	120	24:09:47	2738	0	0	0	0
Rd:34	0	0:00:00	0	5	24:32:53	2939	0	0	0	0
Rd:35	147	19:11:21	2215	7	3:52:13	460	150	2	5	0
Rd:37	1447	15:18:23	1452	4	0:01:29	1	302	11	76	0
Rd:38	1236	18:28:33	1538	1	25:41:00	41	160	0	37	0
Rd:39	78	3:21:11	336	141	21:26:04	2416	343	255	0	0
Rd:42	56	15:01:09	1748	122	5:38:42	350	3	0	0	0
Rd:43	0	0:00:00	0	3	25:35:32	3070	0	0	0	0
Rd:44	0	0:00:00	0	4	25:22:22	3038	0	0	0	0
Rd:45	277	22:35:48	2449	82	2:41:01	197	86	0	35	0
Rd:47	882	21:25:50	2023	4	0:01:05	0	470	58	16	0
Rd:48	0	0:00:00	0	13	23:59:23	2872	0	0	0	0
Rd:49	0	0:00:00	0	7	23:58:38	2869	0	0	0	0
Rd:51	348	22:22:04	1386	68	1:26:53	101	271	16	1291	76
Rd:53	339	52:49:00	0	448	2:58:48	0	2	0	0	0
Rd:54	0	0:00:00	0	1	24:59:03	2750	0	0	0	0
Rd:55	22	21:33:46	2555	12	4:02:41	479	788	20	9	0
Rd:56	4	15:18:16	1835	1	8:46:36	1053	35	0	13	0
Rd:58	928	14:59:38	684	12	6:16:52	724	1053	232	87	0
Rd:60	0	0:00:00	0	2	22:19:51	2623	0	0	0	0
Rd:62	0	0:00:00	0	41	24:47:28	2952	0	0	0	0
Rd:64	0	0:00:00	0	1	5:42:33	685	0	0	0	0
Rd:65	0	0:00:00	0	2	25:44:39	3064	0	0	0	0
Rd:68	0	0:00:00	0	6	23:55:35	2821	0	0	0	0
Rd:69	0	0:00:00	0	1	23:40:11	2736	0	0	0	0
Rd:70	0	0:00:00	0	1	26:05:15	2824	0	0	0	0
Rd:71	0	0:00:00	0	1	24:02:05	2853	0	0	0	0
Rd:72	72	3:56:45	442	29	20:08:31	2399	191	0	15	0
Rd:74	11	0:29:00	0	1044	23:32:16	913	0	0	0	0

Rd:75	0	0:00:00	0	1	20:48:17	2477	0	0	0	0
Total			78412			111405	19596	2898	5614	587

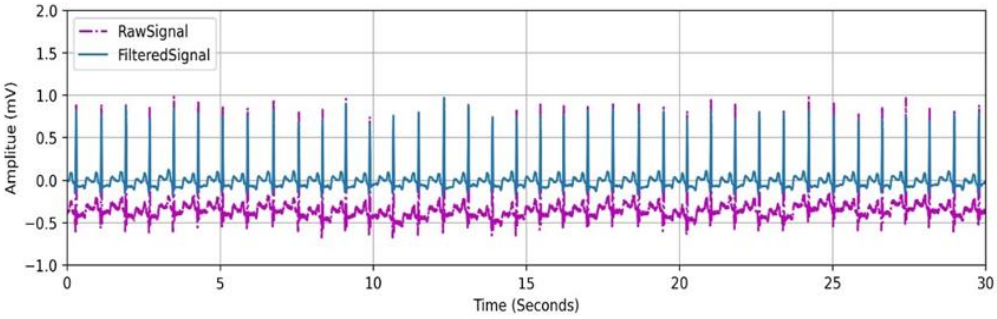
4.1.3 Data Preprocessing

4.1.3.1 Noise Filtering

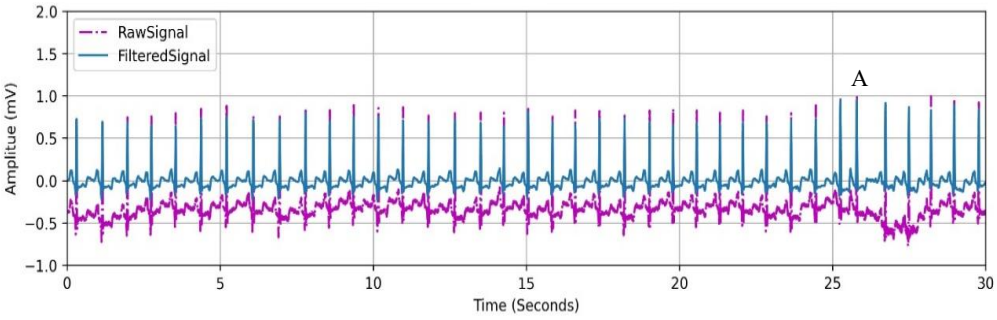
Since the raw ECG signal in the record involves noises such as powerline interferences, baseline wanders, and electromyographic noises, all the selected 30-second time-series signals are firstly filtered using 5th order high-pass Butterworth filter with a cutoff frequency of 0.5 Hz to eliminate the baseline wander. Then, the powerline interference of 60 Hz is removed from each segment with the use of a band stop filter centered at 60 Hz since the original raw signal include 60 Hz powerline interference. The comparison between the signals before and after filtering can be seen in Fig. 4.11 for each ECG type.



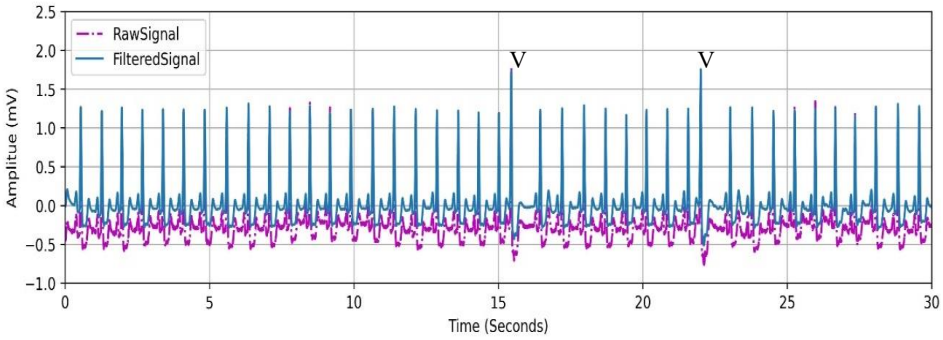
(a)



(b)



(c)



(d)

Fig 4.11. Comparison of Raw and Filtered Signal for (a) AF (b)NSR (c)PAC and (d) PVC

4.1.3.2 Data Transformation (Image Creation from 1D 30-second Segments)

After filtering, the heart rate is calculated with the Neurokit2 algorithm [51], which firstly detects QRS complexes from the steepness of the absolute gradient of the 30-second ECG signal record. Subsequently, R-peaks are then selected from local maxima within the detected QRS complexes. The calculated heart rate (beat per minute) is then divided by 60 seconds to get the cardiac cycle value as presented in Fig 4.12.

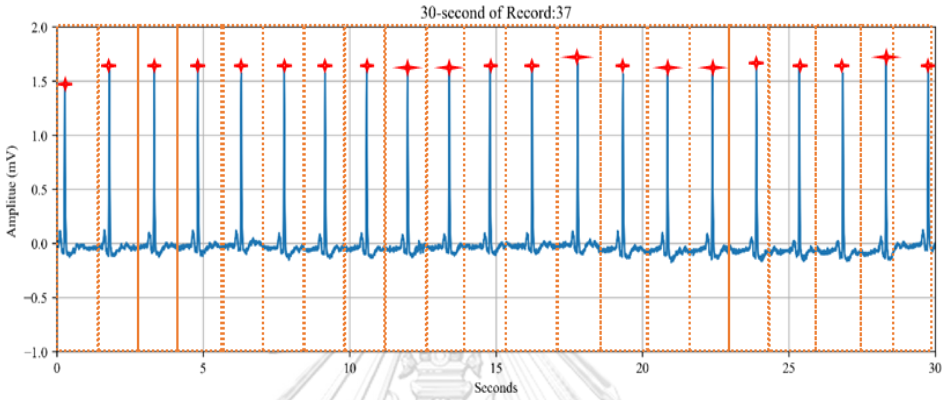


Fig 4.12. Example of 30-second ECG record with detected R-peaks and segmentation per cardiac cycle

The individual beats within 30 seconds segment are then plotted on a plain white background with the obtained cardiac cycle value for the x-axis and the amplitude range of -3 mv to 3 mv for the y-axis. Finally, the plotted beat images are overlayed over each other to form a single synchronized beat image which represents the overall information of all the beats in 30-second of the ECG record as in Fig 4.13.

According to this image transformation approach, the difference between arrhythmias and NSR can be significantly observed. NSR is regular beats composed of noticeable P, Q, R, S and T waves in each beat and complete QRS complex for all beats look like each other. Consequently, the segmented individual heartbeats in the same window frame are synchronized to each other as shown in Fig 4.14 (c). Based on this image, the 30-s record of the ECG signal can be classified as free from any type of arrhythmias with the use of a single image. The benefit of the approached image transformation method becomes noticeable in PAC. As mentioned above, PAC is hiding in the normal heartbeat with complete QRS complex whereas PAC beat is the early appearance of a heartbeat. Since the PAC is not aligned with the normal beat, the appearance of early P-wave and incomplete compensatory beat of PAC appear obviously as in Fig 4.14 (a). Moreover, the ST depression of PVC can be seen clearly within an image as shown in Fig 4.14 (b).

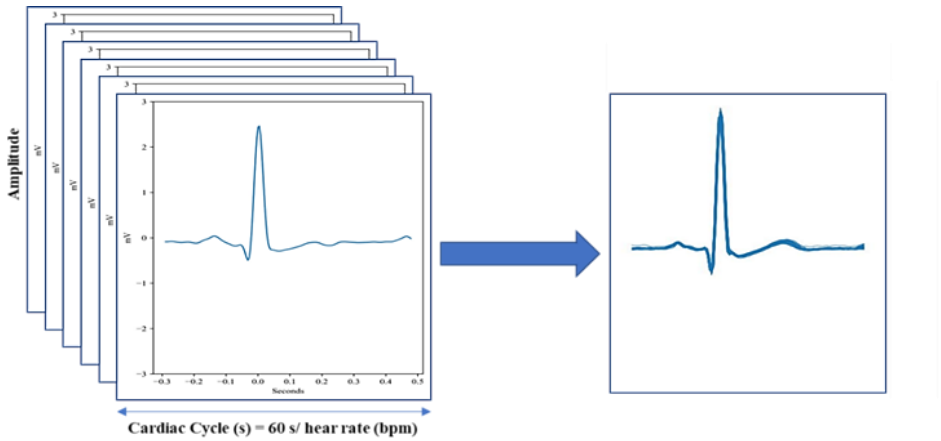


Fig. 4.13. The formation of the overlapped beat image after overlapping individual beat images in a single frame.

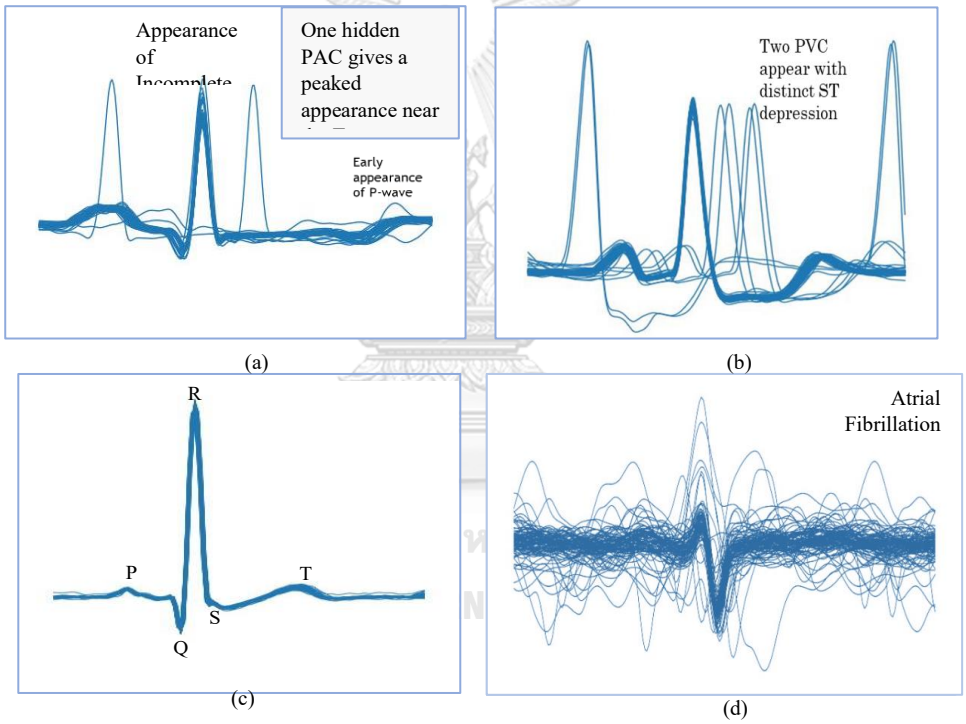


Fig. 4.14. Comparison of Overlapped Images which represents 30-second ECG signals of each Arrhythmias (a) PAC and (b) PVC (c) NSR and (d) AF.

Finally, for AF, the characteristic of AF is irregularly irregular beats with the absence of P-wave and the presence of F-wave. As result, the combined image of beats in AF is totally different from the NSR image shown in Fig 4.14 (d). Since the overall differentiation of AF can be seen in one image, the misdiagnosed of AF from other arrhythmias is also reduced. The proposed image transformation method allows for the classification of different types of arrhythmias and NSR based on visual patterns and features observed in the ECG beat overlapped images.

4.2 Research Procedure

4.2.1 Loading the dataset

The training set is the dataset that the neural networks use to learn any potential underlying patterns or relationships that will enable making predictions. It is needed to be careful that the training set should cover the representative of all the classes without biasing on particular class. If the training set is biased, the overfitting in training may be encountered. To mitigate this risk, the validation dataset is needed to analyze the performance of the model effected by the hyperparameter choices while the model is compiled. Based on comparing the model training graph with training and validation, we can analyze whether the model is overfitting or underfitting. Finally, the test dataset is set up to determine the performance of the trained model on unseen dataset in assessing the model's ability to perform accurately and effectively on data that it was not specifically trained on.

In this research , the datasets outlined in Table 4.2 provide insights into the robustness of the cardiac arrhythmia classification model. These datasets are designed to analyze the model's performance under different criteria, further examining the potential usability of the developed model in medical practice. Analyzing the model's performance across diverse datasets provides valuable insights into its robustness in classifying different types of cardiac arrhythmias, aiding in determining its feasibility and potential utilization in practical healthcare applications for cardiac arrhythmia classification.

In the input pipeline of the model for this research, the dataset is divided into 3:1:1.5 for training, validation and testing respectively as in Table 4.2. Initial step of the model training, the input pipeline for training the model is prepared for achieving high performance and to reduce the time required to execute a single training step by delivering data for the next step before the current step has finished. In this research, prefetching is added into the input pipeline to reduce the extraction time of data of the model from the dataset. This enhances the input pipeline reading the data for the next step while the model is executing current training. Since there are four classes intended to classify in this research, four folders included with a thousand photos are created with respect to their class name. In addition, the images within the data sets are resized to (224x224) and rescaled into the range of [-1,1].

Table 4.2. Datasets Configuration and Class Distribution

DataSets	NSR	AF	PAC	PVC
D1	Pure NSR	Pure AF	PAC>=20%	PAC>=20%
	587	587	587	587
D2	Pure NSR	Pure AF	PAC>=15%	PAC>=15%
	1184	1184	1184	1184
D3	Mixed NSR (150 of PAC<20) (150 of PVC<20) 287 of PureNSR	Pure AF 587	PAC>=20% 587	PAC>=20% 587
D4	Mixed NSR (250 of PAC<15) (250 of PVC<15) 584 of PureNSR	Pure AF 1184	PAC>=15% 1184	PAC>=15% 1184

4.2.2 Setting up the model

In the research transfer learning from the pretrained lightweight neural networks are applied with weights trained on the Image Net Dataset. Therefore, the base models are created and applied as a feature extractor. The lower layers of the models are frozen to prevent updating the weights during training. However, the classification layer (top layer) of the original models are modified in accordance with the research requirements since the pretrained model is designed to classify thousands of classes. The model is complied with the chosen optimizer and loss function before training. The details of the chosen optimizer, learning rates and other parameters are provided in Section 4.2.3.1.

4.2.3 System Implementation

The research with lightweight neural networks is performed from the pretrained model in Python using TensorFlow and Keras which are open-source popular neural network libraries. In this research, the base model is tested with three models : MobileNetV2, MobileNetV3 and EfficientNetB0 which are pretrained with weights from ImageNet Datasets consisting of 1.4M images and 1000 classes. The basic architecture of the neural networks in this research are illustrated in block diagrams in section 2.5.2. This base of trained knowledge will be used to classify our specific cardiac arrhythmias dataset. First, the layer of MobileNet used for feature extraction is set up by specifying the top layer trainable as “False”. The features from the base models converted to single 1280-element vector per image using a

tf.keras.layers.GlobalAveragePooling2D layer. After that, tf.keras.layers.Dense (fully connected) layer is applied to convert these features into probabilities of each class .

4.2.3.1 Experimentation with the Deep Learning Light Weight Neural Networks

In preparation for model training, the dataset is subjected to a preprocessing stage to align it with the specific requirements of the neural network being employed. For instance, when utilizing MobileNetV2 for classification, the dataset is first preprocessed using the keras.application.MobileNetV2.preprocess_input function, which involves performing rescaling and resizing operations. The base model is then established with weights initialized to ImageNet and the classifier activation set to 'softmax', while ensuring that it is not trainable. Subsequently, the pretrained model is loaded with a modified classification top layer tailored to our specific needs. Finally, the model layers are sequenced, considering the number of classes (4 in our case).

The model is constructed by defining the input shape as (224, 224, 3). In addition, data augmentation is applied, incorporating random zooming in or out of the ECG beat images within a specified maximum zoom of 50%. This augmentation technique serves to enhance the model's resilience to variations in scale and appearance, allowing it to effectively analyze and interpret ECG beat images with improved robustness. Then, preprocessing is performed on the inputs using preprocess_input. The base model is applied to the preprocessed inputs with the training parameter set to False, thereby leveraging the features learned from the pretrained model. Global average pooling is then applied to obtain a fixed-length vector representation. Batch normalization and dropout layers are added to enhance the model's performance utilizing the GlorotUniform initializer for weight initialization.

Subsequently, the model is compiled with the Adam optimizer, using Xavier as the weight initializer and Zeros as the bias initializer as in Table 4.3. The learning rate is set to 0.001, and a mini-batch size of 200 is employed. During training, the model's performance is evaluated using the SparseCategoricalAccuracy metric. Once the model has been trained, its performance is evaluated on the test dataset. Moreover, additional experimentation is performed by adjusting various hyperparameters, as outlined in detail in Section 5.1.2. This includes exploring different optimizers, incorporating batch normalization, and introducing dropout layers.

Table 4.3 Experimental Configuration for Model Training

Optimizer	Weight Initializer	Bias Initializer	Learning Rate	Mini Batch Size	Performance metrices
Adam	Xavier	Zeros	0.001	200	SparseCategorical Accuracy

4.2.4 Evaluation of the Classification Model Effectiveness

The performances of the applied models are evaluated using six standard performance measures such as sensitivity, specificity, accuracy, precision, recall and f1_score. These values are enumerated based upon the true positive (TP) , true negative (TN) , false-negative (FN) and false-positive (FP) from the confusion matrix.

4.2.4.1 Confusion matrix

Confusion matrix is a table applied in classification problems for both binary and multiclass classification problems to analyze the outcomes of the predictions of the model. The rows represent the true value of the classes whereas the columns represent the predicted outcomes. Based on the number of classes (N) to do classification, the confusion matrix becomes N x N matrix. In this research, the resulted confusion matrix of the models is 4 x 4 matrix since 4 classes (AF, NSR, PAC and PVC) are classified by the models. The confusion matrix, applied to a dataset consisting of four classes of cardiac arrhythmias is illustrated in Figure 4.15. Based on this confusion matrix, True Positive(TP),False Positive(FP),True Negative(TN) and False Negative(FN) for each class can be recognized so that the way of the prediction's correctness or inaccuracy can be aware. From these values , standard performance measures such as sensitivity, specificity, precision, recall, f1-score, and accuracy can be calculated.

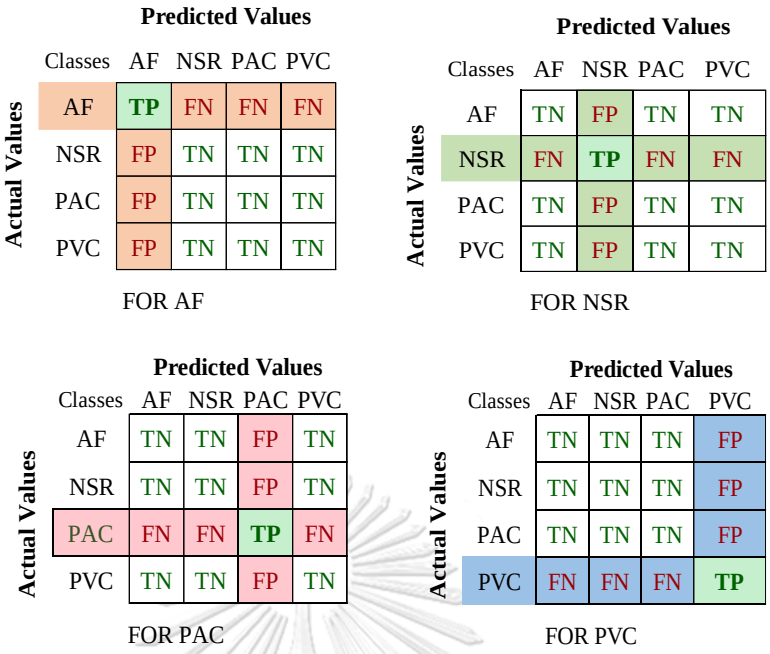


Fig 4.15 Semantic of the Confusion Matrices for Four Classes (AF, NSR, PAC and PVC) result

4.2.4.2 Sensitivity / Recall

Sensitivity is the indicator of the model which correctly classified the positive of the class, known as true positive rate. The calculated formula of the sensitivity of the model is as :

Sensitivity (Recall) =TP/(FN+TP)

4.2.4.3 Specificity

Specificity focuses on correctly classified the negative of the class by the model, known as true negative rate. The formula for calculating specificity is as follow:

Specificity=TP/(TN+FP)

4.2.4.4 Accuracy

Accuracy value indicates the number of correctly predicted (true positive and true negative) classes by the model. The accuracy is calculated as per follow:

Accuracy=(TP+TN)/(TP+TN+FP+FN)

4.2.4.5 Precision

The Precision, also known as Positive Predictive Value (PPV) shows how precisely the model can predict the positive value of the classes. The formula for precision is as below:

Precision=TP/(TP+FP)

4.2.4.6 F1_score

F1_score represents the harmonic mean of the combination of precision and recall. The formula for f1-score is presented below:

$$\text{F1\_score} = \frac{2}{\frac{1}{\text{Precison}} + \frac{1}{\text{Recall}}}$$

4.2.5 Experimental Environment

This research is performed to train the deep learning model with personal computer in which specifications are as follow:

Device name :	Dell Vostro 5490
Processor :	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
Graphic :	NVIDIA GeForce MX250
Installed RAM :	16.0 GB (15.8 GB usable)
System Type :	64-bit operating system, x64-based processor
Operating System:	Windows11

After training, the model is converted to the requisite format for deployment on mobile devices. In order to substantiate the potential of lightweight deep learning for deployment on mobile devices and assess the model's performance, an Oppo Reno 4 mobile phone is employed as the experimental platform. The device is powered by a Qualcomm Snapdragon 720G Octa-core processor, operates on Color OS version 12.1, and is equipped with 8 GB of RAM. By utilizing this specific configuration, a comprehensive evaluation of the lightweight deep learning approach can be conducted, ensuring reliable and accurate results. This choice of hardware provides an appropriate setting for the examination and validation of the claim regarding the suitability and effectiveness of lightweight deep learning in the context of mobile devices.

5 Results and Discussion

5.1 Experimental Results from Testing with Different Scenarios

5.1.1 Experimentation for Binary Classification (AF and NSR) with Lightweight Deep Learning Neural Networks

The main objective of this research is to accurately detect atrial fibrillation (AF) and therefore, the initial focus is on performing a binary classification of AF and NSR (normal sinus rhythm). The images used in the experiments are resized to a dimension of 224x224, which is compatible with the model being employed. Data augmentation techniques, such as random magnification, and a normalizing layer are applied to the input pipeline. These techniques aim to improve the performance and diversity of the ECG database used for training the model. Furthermore, the last three layers of the model, namely the global average pooling 2D layer, dropout layer, and prediction layer, are modified. This modification is necessary to adapt the model specifically for binary classification, as required by the study.

Furthermore, the optimal image quality required for detecting arrhythmias in resource-constrained environments is investigated. The overlapped beat images are examined under three different scenarios: original images without compression, images compressed to 50% of the original quality, and images compressed to 10% of the original quality using the widely used JPEG compression algorithm. The resulting compressed images had peak signal-to-noise ratios (PSNR) of 42.42 dB for the 50% compressed images and 33.55 dB for the 10% compressed images. Notably, the file sizes of the JPEG-encoded images were approximately 36 KB for the original ECG image, 12 KB for the 50% compressed image, and a mere 5 KB for the 10% compressed image.

While evaluating the performance, the original uncompressed images exhibited impressive results, with a sensitivity of 98.1%, specificity of 99.6%, and an overall average accuracy of around 99%. Interestingly, even when the images were compressed by 50%, the sensitivity remained consistently high at 98.3%, while the specificity showed a slight decrease to 98.8%. Consequently, the overall average accuracy remained commendable at around 98%. For the 10% compressed images, a minor reduction in sensitivity is observed, resulting in a value of 95%, along with a specificity of 97.1%. Despite this reduction, the overall average accuracy remained notable at around 96%.

These findings implied that image compression has minimal impact on the effectiveness of arrhythmia detection. The sensitivity, specificity, and overall accuracy

metrics remained robust, even with the compressed images. Consequently, these results suggest that employing compressed images is a practical and viable approach in resource-constrained settings, without compromising the performance of arrhythmia detection.

Continuously, the performance of various models in detecting AF and NSR is evaluated, and the results are summarized in Table 5.1. The models, including "EfficientNetB0," "MobileNetV2," and "MobileNetV3," are tested using original ECG images and compressed images at 50% and 10% of the original quality. The average accuracy and training time for each model and image configuration are presented in the table. According to the table, the performance evaluation of three models, "EfficientNetB0," "MobileNetV2," and "MobileNetV3," for atrial fibrillation (AF) and normal sinus rhythm (NSR) detection revealed robust performance. The models achieved high accuracy levels, with average accuracies of 99%, 99%, and 99% respectively when trained and tested on original ECG images. Even with image compression to 50% and 10% of the original quality, the models maintained strong accuracy levels, with average accuracies ranging from 98% to 97%. The training times for the models varies significantly, with "EfficientNetB0" taking the longest time while "MobileNetV3" demonstrated the shortest training duration. These variations in training times are important factors for this research when selecting a model for deployment, considering the available computational resources and time constraints.

Table 5.1. Performance and Training Times of Different Models for AF and NSR Classification

For only AF and NSR													
Model	**Input Image**	**First**	**Second**	**Third**	**Fourth**	**Fifth**	**Sixth**	**Seventh**	**Eighth**	**Ninth**	**Tenth**	**Avg Accuracy**	**Training Time**
Efficient NetB0	Original	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.98	0.99	0.99	1d 15h 11min 27s
	50%	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	
	10%	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	
Mobile NetV2	Original	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	8h 44min 25s
	50%	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	
	10%	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	
Mobile NetV3	Original	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98	0.99	0.98	0.99	3h 57min 5s
	50%	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	
	10%	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	0.96	

5.1.2 Experimentation for Multiple Classification (AF, NSR, PAC and PVC) with Lightweight Deep Learning Model

The scope of this research expands beyond the detection of atrial fibrillation (AF) to investigate the applicability of our proposed data transformation method in classifying other cardiac arrhythmias, specifically premature atrial contractions (PAC) and premature ventricular contractions (PVC). PAC and PVC are ectopic-beat type arrhythmias that, although distinct from AF, have a high likelihood of progressing to AF and pose serious threats to life in severe cases [23, 25, 52]. Consequently, we proceeded with an investigation of model performance in classifying these four cardiac arrhythmias using the overlapped ECG beat images generated through our proposed data transformation method.

The findings from table 5.2 and 5.3 demonstrate the significance of employing diverse scenarios and techniques in the classification of cardiac arrhythmias using deep learning models. In this research, three deep learning models, namely MobileNetV2, MobileNetV3, and EfficientNetB0, are analyzed using various scenarios and performance metrics. These models are trained and tested on a dataset comprising different types of cardiac arrhythmias, including AF, NSR, PAC, and PVC. Each scenario utilized distinct techniques such as transfer learning, data augmentation, and adjustments in dropout rates and batch normalization. The tables provide detailed evaluation metrics, encompassing sensitivity, specificity, precision, F1 score, and accuracy, for each model and scenario.

Firstly, MobileNetV2 demonstrated exceptional performance in the transfer learning scenario for all arrhythmia classes, with sensitivities ranging from 0.92 to 0.97 and specificities from 0.97 to 0.99. Furthermore, the implementation of the data augmentation technique resulted in further improvements, leading to higher sensitivity and specificity values in most cases. The impact of additional experiments involving dropout regularization and batch normalization varied, resulting in diverse effects on the model's performance.

In contrast, MobileNetV3 exhibited relatively lower performance in the transfer learning scenario compared to MobileNetV2, with sensitivities ranging from 0.72 to 0.98 and specificities ranging from 0.91 to 0.99. However, the application of data augmentation techniques significantly enhanced the results, leading to higher sensitivities and specificities. Furthermore, the introduction of dropout regularization and batch normalization further improved the model's performance in most cases.

Finally, EfficientNetB0 exhibited competitive performance in the transfer learning scenario, with sensitivities ranging from 0.82 to 0.98 and specificities ranging from 0.96 to 0.99. The implementation of data augmentation techniques resulted in slight improvements in sensitivity and specificity. Similarly, to the other models, the impact of incorporating dropout regularization and batch normalization varied, influencing the overall performance. The training and validation graph are provided at APPENDIX D as an example for MobileNetV3.

Table 5.2. Performance Evaluation of Each Model for different Scenario

Model	Scenerio	Sensitivity				Specificity				Precision				f1_score				Accuracy
		AF	NSR	PAC	PVC	AF	NSR	PAC	PVC	AF	NSR	PAC	PVC	AF	NSR	PAC	PVC	
Mobile NetV2	1	0.92	0.95	0.95	0.97	0.97	0.98	0.99	0.98	0.92	0.95	0.99	0.96	0.92	0.95	0.97	0.97	0.95
	2	0.95	0.93	0.96	0.97	0.97	0.98	0.96	0.98	0.94	0.97	0.94	0.97	0.94	0.95	0.95	0.97	0.95
	3	0.93	0.96	0.95	0.95	0.98	0.99	0.96	0.99	1.00	1.00	0.91	0.96	1.00	0.95	0.93	0.96	0.97
	4	0.95	0.96	0.97	0.95	0.98	0.99	0.96	0.99	0.98	0.98	0.93	0.96	0.96	0.97	0.95	0.95	0.96
	5	0.97	0.91	0.94	0.98	0.96	0.99	0.98	0.98	0.91	0.98	0.94	0.96	0.94	0.94	0.94	0.97	0.95
	6	0.97	0.98	0.97	0.98	0.98	0.99	0.99	0.98	0.96	0.99	0.98	0.96	0.96	0.98	0.97	0.97	0.97
	7	0.97	0.96	0.97	0.97	0.98	0.99	0.97	0.99	1.00	1.00	0.99	0.98	0.96	0.98	0.98	0.97	**0.98**
Mobile NetV3	1	0.72	0.91	0.78	0.98	0.96	0.91	0.95	0.96	0.89	0.79	0.81	0.91	0.80	0.85	0.80	0.95	0.85
	2	0.91	0.99	0.85	0.95	0.96	0.97	0.97	0.99	0.91	0.92	0.91	0.97	0.91	0.95	0.88	0.96	0.93
	3	0.93	0.99	0.93	0.96	0.98	0.98	0.98	0.98	0.95	0.96	0.94	0.96	0.94	0.97	0.94	0.96	0.95
	4	0.93	0.91	0.97	0.97	0.98	0.99	0.96	0.98	0.95	0.97	0.91	0.96	0.94	0.94	0.94	0.96	0.95
	5	0.94	0.93	0.97	0.97	0.98	0.99	0.96	0.99	0.96	0.98	0.92	0.97	0.95	0.95	0.94	0.97	0.95
	6	0.99	0.96	0.99	0.97	0.99	0.99	0.98	0.99	0.98	0.99	0.95	0.98	0.98	0.97	0.97	0.97	0.97
	7	0.99	0.96	0.99	0.98	0.99	0.99	0.98	0.99	0.98	0.99	0.96	0.98	0.98	0.97	0.97	0.98	**0.98**
Efficient NetB0	1	0.82	0.96	0.93	0.98	0.96	0.98	0.99	0.96	0.99	0.94	0.98	0.89	0.85	0.95	0.96	0.93	0.93
	2	0.85	0.95	0.95	0.97	0.96	0.97	0.98	0.97	0.91	0.96	0.94	0.92	0.88	0.95	0.94	0.94	0.94
	3	0.89	0.95	0.95	0.97	0.97	0.97	0.98	0.97	0.93	0.96	0.94	0.92	0.91	0.95	0.94	0.94	0.94
	4	0.93	0.97	0.96	1.00	0.97	1.00	1.00	0.97	0.93	1.00	1.00	0.90	0.93	0.98	0.98	0.95	0.96
	5	0.96	1.00	0.93	1.00	1.00	0.98	1.00	0.97	1.00	0.96	1.00	0.92	0.98	0.98	0.96	0.96	0.97
	6	0.99	0.97	1.00	0.98	0.99	0.99	0.98	0.99	0.99	0.99	0.96	0.99	0.99	0.98	0.98	0.98	0.98
	7	0.98	0.98	0.98	0.99	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	0.98	0.99	0.98	0.99	**0.99**

Table 5.3. Different Scenarios for Testing the Performance of the Models

No.	1	2	3	4	5	6	7
Scenerio	Transfer Learning	Data Augment	Data Augment Dropout(0.1)	Data Augment Dropout(0.2)	Data Augment Dropout(0.4)	Data Augment Dropout(0.2) BatchNormalized	Data Augment Dropout(0.4) BatchNormalized

5.1.3 Experimentation for Different Training Datasets

After conducting experiments with different hyperparameters in Section 5.1.2, we performed further experiments using the optimized hyperparameters. In this section, we examine how the models perform on test datasets when trained with different datasets. The experimental setup for training the models remains the same as described in Section 4.2.3.1, but we adjusted the hyperparameters to reflect the optimized values obtained in Section 5.1.2. These optimized values are presented in Table 5.3, and the models are trained using diverse datasets listed in Table 4.2. This experimentation allows us to evaluate the models' performance on various datasets and determine how effective the optimized hyperparameters are in improving classification accuracy.

Out of the three models considered, MobileNetV3 is the preferred choice due to its lightweight design, characterized by fewer parameters and shorter training duration. However, it is worth noting that MobileNetV3 is sensitive to variations in the training data. Therefore, in the subsequent section, our focus shifts to assessing the model's performance under diverse experimental conditions using the MobileNetV3 architecture. This analysis aims to provide valuable insights into the model's suitability and its adaptability.

Across these datasets, the model consistently achieves high sensitivity values, ranging from 0.97 to 0.99 except the model trained with D3 as shown in Fig 5.1, accurately identifying each class when they are tested with simple test dataset in which NSR data is not interrupted with PAC and PVC less than 20%. Similarly, the specificity values consistently range from 0.97 to 0.99, indicating the model's ability to correctly classify instances not belonging to each class. The F1-scores, which consider both precision and sensitivity, range from 0.95 to 0.98, demonstrating a balanced performance for each class. Overall, the MobileNetV3 model exhibits reliable accuracy in classifying the four classes accurately across different training datasets.

In addition, we retested the models with datasets that included mixed NSR records containing occasional PAC and PVC instances below the 20% threshold. This was done to enhance the relevance and practicality of the test data and to provide a comprehensive assessment of the model's performance in real-life scenarios where normal sinus rhythm records may contain such instances. The resulting performance indicators are shown in Figure 5.2. Comparing the assessment of the models presented in Fig 5.1 and Fig 5.2 demonstrates impressed good accuracy achieved, ranging from 0.93 to 0.9, across all models tested. Notably, these models exhibit superior performance when subjected to a dataset exclusively comprising instances of Pure NSR, AF, PAC>=20%, and PVC>=20%.

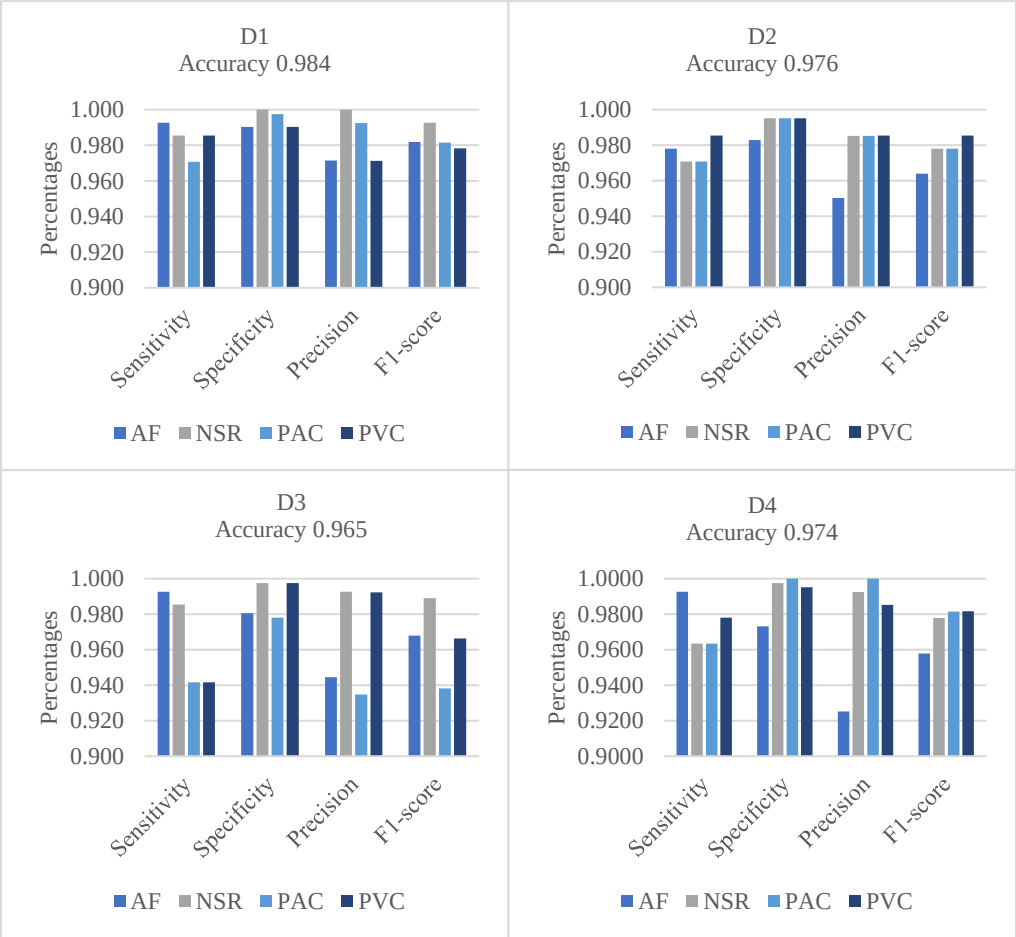


Fig 5.1. Performance Comparison of Models Trained with Four Datasets Tested on Test Data Set including Pure NSR, AF, PAC>=20%, and PVC>=20%

These results signify the effectiveness of the models in accurately classifying these specific types of arrhythmias. However, an important observation is that models trained solely on Pure NSR data experience a significant decrease in precision scores when tested with a dataset containing mixed NSR records, which include instances of PAC and PVC below the 20% threshold. This decline implies a higher likelihood of false-positive classifications for NSR records in the presence of limited PAC and PVC involvement. In other words, these models are more prone to misidentifying NSR records as arrhythmias, despite the records only having minimal representation of PAC and PVC.

In contrast, our findings reveal that models trained on a dataset encompassing both pure NSR and mixed NSR records with minimal PAC and PVC involvement maintain precision and F1-score values above the 0.95 threshold. This particular model demonstrates a notable ability to distinguish between NSR records with minimal PAC and PVC representation and those that genuinely represent pure NSR, leading to improved precision and overall performance.

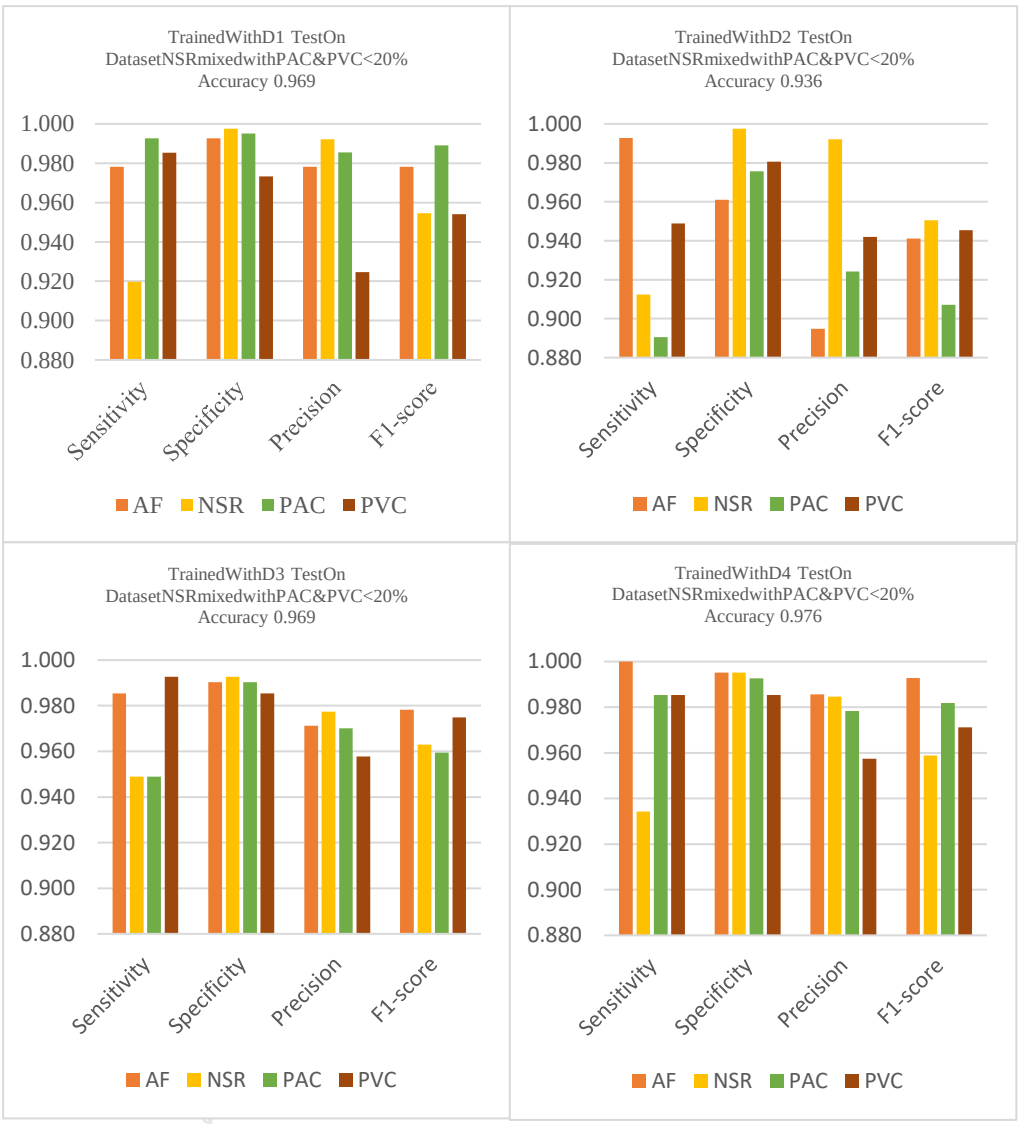


Fig 5.2. Performance Comparison of Models Trained with Four Datasets Tested on Test Data Set including NSR*(Pure and Mixed with PAC&PVC<20%)*, AF, PAC>=20%, and PVC>=20%

In summary, our experiment in this section highlight the significance of using specific data types during model training, particularly when classifying NSR records mixed with a few PAC and PVC instances. These results suggest that using datasets that include both Pure NSR and mixed NSR records can help improve the accuracy and performance metrics like precision and F1-score. This approach is particularly beneficial when dealing with situations where NSR records might occasionally have a few PAC and PVC instances below a specific threshold.

On the other hand, correctly identifying NSR records with occasional PAC and PVC instances provides healthcare providers with confidence in confirming normal heart function, reducing the necessity for further investigations, and offering reassurance to patients. Accurate classification of NSR records is therefore crucial for making appropriate medical decisions and optimizing patient care in the context of arrhythmia detection and diagnosis.

From an engineering standpoint, accurately classifying NSR (normal sinus rhythm) records that occasionally contain a small number of PAC (premature atrial contraction) and PVC (premature ventricular contraction) instances below a specific threshold is crucial for achieving robust and reliable classification of AF (atrial fibrillation). This is because AF detection heavily relies on distinguishing it from NSR. Misclassifying NSR records with occasional PAC and PVC instances as AF could lead to false positives and inaccurate diagnosis, which can have significant implications for patient management and treatment decisions. Therefore, engineering-wise, ensuring precise classification of NSR records is essential for developing a highly accurate AF detection system.

5.2 Deploy Model on Android (using kotlin language constructed in android studio)

Along with the rise of mobile devices and the popularity of machine learning, the interest in deploying machine learning models on these devices for offline use and real-time inference has also been increased. By deploying machine learning models on Android devices, it becomes helpful in scenarios where internet connectivity is limited or unreliable, such as in remote areas or during natural disasters. In particular, deploying machine learning models on Android devices has become a popular research area due to the widespread use of Android devices and the availability of powerful hardware. Applications on mobile devices can provide real-time inference without requiring an internet connection or cloud infrastructure.

The application utilizes a pre-trained TensorFlow Lite model to classify electrocardiogram (ECG) images obtained through the device's camera or from the gallery. This section explores the technical aspects, implementation, and the challenges encountered during its development. The Android application's system architecture and implementation details are also presented in this section. The deployment process involves downloading the Android Studio application from the website https://developer.android.com/studio, according to the specific operating system setup. As the application is designed for mobile phones or tablets, it is important to select the appropriate initial layout option for these devices. The development of the application entails the utilization of the Kotlin programming language, for its user-friendly coding interface that promotes the efficient implementation of desired

functionalities. This language's built-in ease of setup and coding enhances the overall ease in developing of the application.

In Figure 5.3, a step-by-step process block diagram is presented, illustrating the deployment of a model and the access to media files for the purpose of classifying cardiac arrhythmias. This diagram provides a comprehensive overview of the procedures involved in efficiently utilizing the model to analyze and categorize abnormalities in cardiac rhythm. Initially, it is necessary to ensure the availability of images that are to be classified by the program. Thus, the images intended for classification must be present within the gallery of the mobile device. By adhering to this diagram, users are guided through the necessary steps to grant access and navigate the procedures required to retrieve media files. This access enables the accurate classification of cardiac arrhythmias utilizing the embedded trained model.

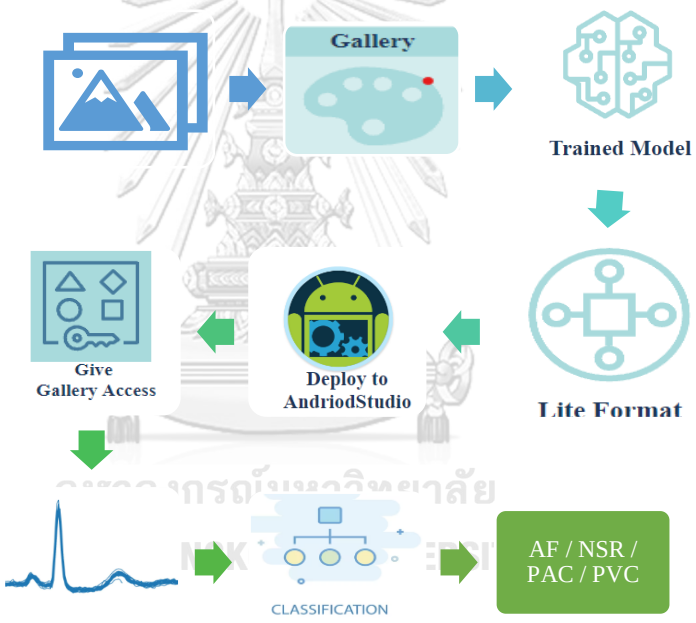


Fig 5.3. Process Block Diagram for Model Deployment and Media File Access in Cardiac Arrhythmia Classification

The application leverages the TensorFlow Lite framework to deploy a pre-trained deep learning model specifically designed for image classification. Prior to deployment, the trained deep learning neural network model needs to be converted into a TensorFlow Lite format, inclusive of assigned weights and bias values. This converted model is then saved as a ".h5" format, which is compatible with Android devices for efficient deployment. Subsequently, the transformed format of the model is embedded into the Android Studio application. By

granting the user access to the image gallery, the model can retrieve the image that the user intends to classify.

Ultimately, the developed classifier application successfully detects occurrences of cardiac arrhythmias. The application, though not commercially available at present, stands as a proof-of-concept in the feasible deployment of lightweight deep learning neural networks on mobile devices as shown in Fig 5.4. The central objective of this study is to showcase the immense potential of deep neural networks for real-world applications that can be executed locally on mobile devices without the need for internet connectivity via cloud platforms. Finally, we also tested the application by deploying MobileNetV2, MobileNetV3 type and EfficientNetB0 type. When those models are transformed into the Tflite format for deployment, there may not be significant differences in performance. This is because the Tflite format is specifically optimized for mobile and resource-constrained devices, ensuring efficient inference and compatibility with various platforms.

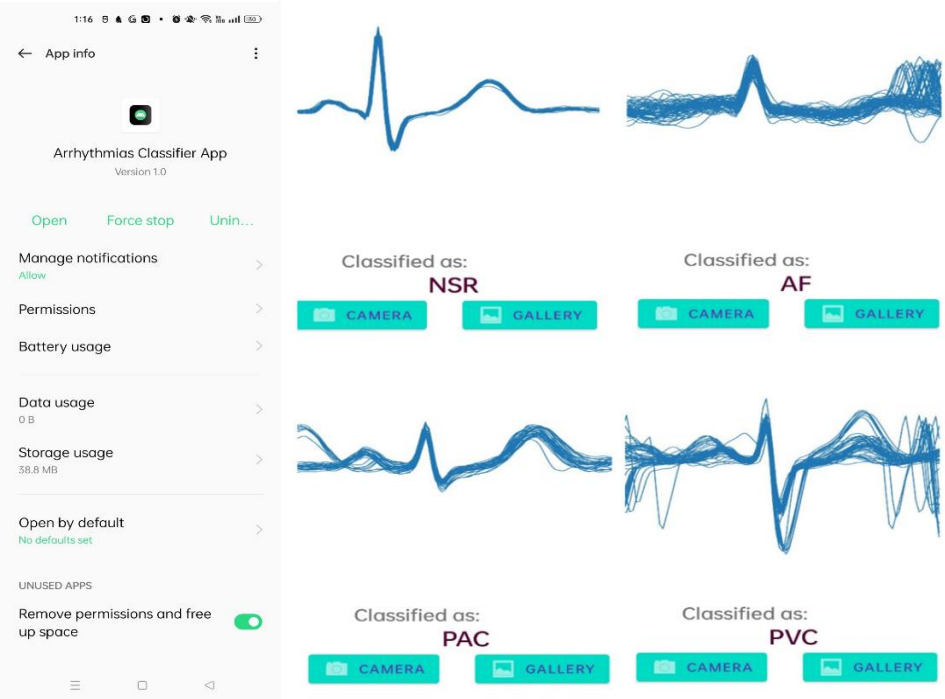


Fig 5.4. The successful results of cardiac arrhythmias classification done by the demo application

In Fig 5.5 of showing confusion matrix from the deployed models, MobileNetV2 (MV2), MobileNetV3 (MV3), and EfficientNetB0 (EfB0), all achieved effective classification of AF and NSR instances, which are essential for cardiac arrhythmia detection. EfB0 exhibited slightly higher accuracy in classifying AF and NSR compared to MV2 and

MV3. Additionally, EfB0 and MV3 showed improved performance in classifying PAC and PVC instances compared to MV2. However, some misclassifications were observed in distinguishing PAC and PVC instances for all models.

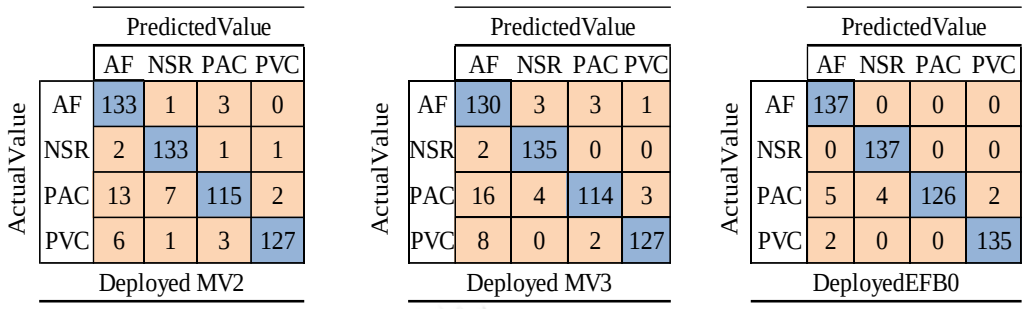


Fig 5.5 Confusion Matrix for Testing the Classification with Deployed Models

Based on the results from Figure 5.5, both the MobileNetV2 and MobileNetV3 applications achieved an accuracy of approximately 0.92. However, there were high false negative values for the PAC class. In the case of the EfficientNetB0 deployment, missed PAC classifications persisted, although the overall accuracy remained around 0.97. When comparing the performance of MobileNetV3 deployed on a mobile device to the trained model of MobileNetV3 on a PC, there was a reduction in performance by approximately 6 percent.

The drop in performance when using a trained model on a mobile phone is due to the limited computing power and memory of mobile devices compared to PCs or servers. These limitations can make it challenging for the model to process and classify efficiently, resulting in lower performance. To make models suitable for mobile use, it is necessary to employ various optimizations and parameter quantization. Therefore, there is a lot of room for future research to explore and improve these optimization methods. Further investigation can focus on finding the right balance between reducing the model's size and maintaining its accuracy. In the future, researchers can concentrate on developing new ways to enhance models on mobile devices. They can create better optimization methods and explore advancements in hardware to improve the efficiency and effectiveness of models on mobile platforms.

6 Conclusion

This study makes two primary contributions to the field of cardiac arrhythmia detection. The first contribution lies in the development of an unconventional method for transforming electrocardiogram (ECG) data, while the second contribution involves the application of a lightweight deep learning neural network to achieve highly accurate classification of cardiac arrhythmias.

As contrary to previous studies that rely on spectrogram or recurrent plot transformations, this research introduces a novel approach for transforming raw ECG data. The proposed method involves converting the time series ECG data into overlapped ECG beat images. This noble transformation allows for a comprehensive visualization of the ECG signal, capturing the intricate details and patterns necessary for accurate arrhythmias classification. Each resulting image is then appropriately labeled according to its corresponding cardiac arrhythmia type, creating a labeled dataset for model training, validation, and testing.

The image dataset of cardiac arrhythmias (atrial fibrillation, normal sinus rhythm, premature atrial contractions, and premature ventricular contractions) is split into 3:1:1.5 for training, validation and testing respectively. The performance of the model is compared across different scenarios to provide a comprehensive evaluation of the model's accuracy , effectiveness, and robustness. The performance of the proposed models has been thoroughly analyzed by optimizing the hyperparameters of the model. This includes the incorporation of batch normalization, dropout, and optimizer techniques. These techniques were utilized to enhance the accuracy of the model and optimize its overall performance. The results demonstrate that the proposed model achieved comparable performance to the existing methods [19-21]. These findings suggest that the proposed model has the potential to be a useful approach in accurately classifying different types of cardiac arrhythmias.

Notably, the results highlight the model's ability to successfully distinguish atrial fibrillation (AF) from other arrhythmias, including normal sinus rhythm (NSR), premature atrial contractions (PAC), and premature ventricular contractions (PVC), using a simple ECG data transformation which is overlapped ECG beat images. According to the results, this classification of cardiac arrhythmias with lightweight neural network showed promising results. Our concept of overlapping ECG beats into a single image has been validated to effectively distinguish AF features from ectopic beat-based arrhythmias like PAC and PVC. This finding contradicts the claim made in a previous study [21] that their research was not

applicable for AF classification. Futhermore, it provides a comprehensive view of the ECG data, enabling easier differentiation between various arrhythmias and normal rhythm.

The applied lightweight neural networks in this reseach are consturcted based on essential MBconv blocks. For MobileNetV2, its characteristics is applying inverted residuals and linear bottlenecks convolutional blocks. For MobileNetV3, it modifies the MobileNetV2 convolutional blocks by adding squeez and extiation module. For EfficientNet B0, it adjusts the MobileNetV2 convolutional blocks size based on the input size, weight and resolution. Based upon their architecutre, the light weight neural networks are beneficial to extract only the manifold of interest (MOI) of the image.

The effectiveness of employing lightweight neural networks as feature extractors to detect cardiac arrhythmias with high accuracy is thoroughly investigated in this research. This approach not only demonstrates superior performance compared to existing methods but also holds promise in handling the memory usage issues of portable devices. As evidence, the trained model has been successfully deployed on an Android mobile phone, further validating its practical application. Based on the obtained results, our approach demonstrated high sensitivity and specificity, achieving an overall accuracy of approximately 0.98 in the classification of all four targeted cardiac arrhythmias. This significant outcome holds potential for future application in home-based ECG screening tests. With its satisfactory performance, this novel approach can lead to the way for the development of a portable cardiac arrhythmia detection system that can be commercially utilized for medical purposes in the future.

However, it is important to acknowledge certain limitations of the study. Firstly, the focus on patient-specific datasets may limit the generalizability of the findings. Incorporating larger and more diverse datasets in future studies would enhance the model's ability to generalize across a broader patient population.

Another limitation is the data preprocessing step, performed on a computer instead of directly on the mobile phone. This presents an opportunity for improvement by refining the preprocessing stage and embedding it into the mobile device itself. Such a development would enhance overall performance, accuracy, and user convenience, allowing users to perform preprocessing and classification on their mobile phones.

Recognizing these limitations provides insights for future investigations. Addressing them through the inclusion of diverse datasets and refining preprocessing techniques while embedding them into mobile devices will contribute to the development of a more effective and user-friendly cardiac arrhythmia detection system.

APPENDICES

APPENDIX A

DATASET PREPARATION

The dataset preparation program performs collecting 30-second ECG signals from the original 24-hour long ECG records in the long-term atrial fibrillation database. Firstly, the required libraries are imported into the program.

```
import os
import wfdb
import itertools
import numpy as np
import pandas as pd
from collections import Counter
```

All the files within the database are loaded into the program.

```
def find_files_with_extension(path=None, extension="atr"):
   files = [os.path.join(
            path,
            name) for name in os.listdir(path) if name.find(extension) != -1]
   return files
files=find_files_with_extension(path="D:\\ECG DB\\long-term-af-database-
1.0.0\\files\\",extension="atr")
file_list=[files[i][:-4] for i in range(len(files))]
```

The objects needed to apply in the program are constructed. In this program, four objects are constructed namely Record, RecordReader , Window and Converter.

```
class Record:

   def __init__(self, parent, signal, symbol, aux, sample):
      self.__parent = parent
      self.__signal = signal
      self.__symbol = symbol
      self.__aux = aux
      self.__sample = sample
      self.__collection = {"signal": self.__signal, "symbol": self.__symbol, "aux": self.__aux,
                           "sample": self.__sample, "hasMissedBeat": self.hasMissedBeat(),
                           "hasUnknownBeat": self.hasUnknownBeat()}

   def __getitem__(self, key):
      return self.__collection[key]
```

```
def __str__(self):
    return "Summary\n" + \
           "Size of signal: " + str(len(self.__signal)) + \
           "\nSize of symbol: " + str(len(self.__symbol)) + \
           "\nSize of aux: " + str(len(self.__aux)) + "\n"

def getInterval(self):
    return (self.__sampfrom, self.__sampto)

def getIndexesOf(self, this=None):
    if (len(self.__symbol) > 0) and (len(self.__aux)) > 0:
        if this=="+":
            return np.where(np.asarray(self.__symbol)=="+")[0]
        elif this == "(N":
            return np.where(np.asarray(self.__aux)=='(N')[0]

def getIntersectOf(self, a, b):
    return np.intersect1d(a,b,return_indices=True)

def getRhythmInterval(self):
    rhythmInterval = list()
    plusIndexes = self.getIndexesOf('+')
    NIndexes = self.getIndexesOf("(N")
    if len(plusIndexes)!=0 and len(NIndexes) != 0:
        _,aIndexes, bIndexes = self.getIntersectOf(a=NIndexes,
                                    b=plusIndexes)
        for i in range(len(bIndexes)-1):
            intervalStart=NIndexes[i]
            intervalEnd=plusIndexes[bIndexes[i]+1]
            interval = (self.__sample[intervalStart],self.__sample[intervalEnd])
            rhythmInterval.append(interval)
        if plusIndexes[-1]==NIndexes[-1]:
            interval = (self.__sample[NIndexes[-1]], len(self.__signal))
            rhythmInterval.append(interval)
    return rhythmInterval

def getValidRhythmInterval(self):
    rhythmIntervals = self.getRhythmInterval()
    return [itval for itval in rhythmIntervals if self.isIntervalValid(interval=itval,
                                                sampling_freq=128, duration=30)]

def isIntervalValid(self, interval, sampling_freq, duration):
    return (abs(interval[1]-interval[0]) >= (sampling_freq * duration))


 def findIndexOfSymbol(self, symbol):
```

```
    if symbol in self.__symbol:
        return np.where(np.asarray(self.__symbol)==symbol)[0]
    return -1

def findQIndex(self):
    return self.findIndexOfSymbol('Q')

def findQuoteIndex(self):
    return self.findIndexOfSymbol("")

def hasUnknownBeat(self):
    return ("Q" in self.__symbol)

def hasMissedBeat(self):
    return ("" in self.__symbol)

def moveToanyQorQuote(self):
    qIndex=self.findQIndex()
    quoteIndex=self.findQuoteIndex()
    moveIndex=qIndex+quoteIndex
    return max(moveIndex)

def moveToNoPAC(self):
    pacIndexes=self.findIndexOfSymbol("A")
    return max(pacIndexes)

def moveToNoPVC(self):
    pvcIndexes=self.findIndexOfSymbol("V")
    return max(pvcIndexes)

def hasPAC(self):
    return ("A" in self.__symbol)

def hasPVC(self):
    return ("V" in self.__symbol)

def getPACPercentage(self):
    pacCount=self.getPACCounts()
    if len(self.__symbol) > 0:
        return (pacCount/len(self.__symbol))*100
    else:
        return 0

def getPVCPercentage(self):
    pvcCount=self.getPVCCounts()
    if len(self.__symbol) > 0:
```

```
            return (pvcCount/len(self.__symbol))*100
        else:
            return 0

    def IsPositive(self, arrType):
        if arrType == "PAC":
            percentage = self.getPACPercentage()
            return ((percentage >=20) and (self.getPVCCounts()==0))
        if arrType =="PVC":
            percentage = self.getPVCPercentage()
            return ((percentage >=20) and (self.getPACCounts()==0))

    def getPACCounts(self):
        return Counter(self.__symbol)['A']

    def getPVCCounts(self):
        return Counter(self.__symbol)['V']

    def which(self):
        return self.__parent


class RecordReader:

    @classmethod
    def read(self, path, number, channel, sampfrom, sampto):
        fullpath = os.path.join(path, number)
        signal = wfdb.rdrecord(fullpath,  sampfrom=sampfrom, sampto=sampto).
p_signal[:,channel]
        ann = wfdb.rdann(fullpath, 'atr', shift_samps=True, sampfrom=sampfrom, sampto=sampto)

        symbol = ann.symbol
        aux = ann.aux_note
        sample = ann.sample

        return Record(parent=number, signal=signal, symbol=symbol, aux=aux, sample=sample)


class Window:

    def __init__(self, width, leftEnd,arrTypeSearchFor, arrTypeNotSearchFor):
        self.width = width
        self.leftEnd = leftEnd
        self.thirtySegments = list()
        self.arrTypeSearchFor = arrTypeSearchFor
        self.arrTypeNotSearchFor = arrTypeNotSearchFor
```

```
        self.rightEnd = self.leftEnd + self.width

    def scan(self, segment):
        while ((self.leftEnd + self.width)<= len(segment["signal"])):
            thirtySeg = self.getRegionOfInterest(segment)
            newLeftEnd = self.checkForUnkownandMissBeat(thirtySeg)
            if  newLeftEnd == None:
                if thirtySeg.IsPositive(self.arrTypeSearchFor):
                    self.thirtySegments.append(thirtySeg)
                    newleftEnd=self.leftEnd+self.width
                    self.leftEnd=newleftEnd
                elif self.arrTypeNotSearchFor in thirtySeg["symbol"]:
                        if self.arrTypeNotSearchFor=="PAC":
                    self.leftEnd=thirtySeg["sample"][thirtySeg.moveToNoPAC()]+distancebetNextbeat
                        if self.arrTypeNotSearchFor=="PVC":
                    self.leftEnd=thirtySeg["sample"][thirtySeg.moveToNoPVC()]+distancebetNextbeat
                else:
                    self.leftEnd = self.leftEnd + distancebetNextbeat
            else:
                self.leftEnd=thirtySeg["sample"][newLeftEnd]+ distancebetNextbeat
        return self.thirtySegments

    def checkForUnkownandMissBeat(self, segment):
        if (not segment.hasUnknownBeat()) or (not segment.hasMissedBeat()):
            newLeftEnd = None
        else:
            newLeftEnd=segment.moveToanyQorQuote()
        return newLeftEnd

    def getRegionOfInterest(self, segment):
        rightEnd = self.leftEnd+self.width
        signal = segment["signal"][self.leftEnd:rightEnd]
        annIndexes=np.intersect1d(np.where(np.asarray(segment["sample"])>=self.leftEnd),
                          np.where(np.asarray(segment["sample"])<=rightEnd))
        symbol=[segment["symbol"][annIndexes[i]] for i in range(len(annIndexes))]
        sample=[segment["sample"][annIndexes[i]] for i in range(len(annIndexes))]
        shifted_sample=[sample[i]-self.leftEnd for i in range(len(sample))]
        aux = [segment["aux"][annIndexes[i]] for i in range(len(annIndexes))]
        return Record(parent=segment.which(), signal=signal, symbol=symbol, aux=aux,
                    sample=shifted_sample)



class Converter:

    def __init__(self, label):
```

```
        self.label = label

    def convert(self, record):
        data = {"signal": [record["signal"]],
                "symbol": [record["symbol"]],
                "sample": [record["sample"]],
                "PAC_percent": [record.getPACPercentage()],
                "PVC_percent": [record.getPVCPercentage()],
                "label": self.label, "record no.": record.which()}
        return pd.DataFrame.from_dict(data)


for i in range(len(record_numbers)):
    recordNumber = record_numbers[i]
    print("At Record : "+str(recordNumber))
    recordPath = "./records"
    sampfrom = 0
    sampto = None
    arrType = "PVC"

    record = RecordReader.read(path=recordPath, number=recordNumber, channel=0,
                               sampfrom=sampfrom, sampto=sampto)
    rhythmIntervals = record.getValidRhythmInterval()
    segments = [RecordReader.read (path=recordPath, number=recordNumber, channel=0,
sampfrom=interval[0], sampto=interval[1]) for interval in rhythmIntervals]
    thirtySegs =[]
    for j in range(len(segments)):
        window = Window(width=3839, leftEnd=0,
                arrTypeSearchFor=arrType, arrTypeNotSearchFor="PAC")
        thirtySegWithinInterval=window.scan(segments[j])
        thirtySegs.append(thirtySegWithinInterval)
    thirtySegs = list(itertools.chain(*thirtySegs))
    con=Converter(arrType)
    thirtySegs_df = [con.convert(seg) for seg in thirtySegs]
    #print(len(thirtySegs_df))
    if len(thirtySegs_df):
        rdDataFrame=pd.concat(thirtySegs_df, axis=0)
        rdDataFrame.reset_index(drop=True)
        rdDataFrame.to_csv("Record "+str(segments[0].which())+" "+arrType+" equaltoORover20
.csv")
```

APPENDIX B

DATA TRANSFORMATION

(FROM 1-DIMENSIONAL ECG SIGNALS INTO 2-DIMENSTIONAL OVERLAPPED IMAGES)

```
import os
import cv2
import json
import shutil
import pathlib
import imageio
import posixpath

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from PIL import Image
from collections import Counter
from datetime import datetime as dt
%matplotlib inline

import wfdb
import neurokit2 as nk
from neurokit2.epochs import epochs_create, epochs_to_df
from neurokit2.signal import signal_rate
from neurokit2 import ecg_peaks

class CSV_Records:
    def __init__(self,path,extension):
        self.path=path
        self.extension=extension
    def RecordPaths_and_RecordNames(self):
        records = [os.path.join(self.path,name) for name in os.listdir(self.path)
                   if name.find(self.extension) != -1]
        record_names=[str(records[i].split(".")[0]).split("\\")[-1] for i in range(len(records))]
        return records,record_names
records=CSV_Records("D:\GPU\data_preparation\khaing\data", 'csv')
files_list,record_names=records.RecordPaths_and_RecordNames()
```

```
def _ecg_segment_window(heart_rate=None, rpeaks=None, sampling_rate=128,
desired_length=None):
    # Extract heart rate
    if heart_rate is not None:
        heart_rate = np.mean(heart_rate)
    if rpeaks is not None:
        heart_rate = np.mean(signal_rate(rpeaks, sampling_rate=sampling_rate,
desired_length=desired_length))
    # Modulator
    m = heart_rate / 70
    # Window
    epochs_start = -0.3 / m
    epochs_end = 0.45 / m
    # Adjust for high heart rates
    if heart_rate >= 90:
        c = 0.15
        epochs_start = epochs_start - c
        epochs_end = epochs_end + c
    return epochs_start, epochs_end

def ecg_segment(ecg_cleaned, rpeaks=None, sampling_rate=128):
    # Sanitize inputs
    if rpeaks is None:
        _, rpeaks = ecg_peaks(ecg_cleaned, sampling_rate=sampling_rate,correct_artifacts=False)
        rpeaks = rpeaks["ECG_R_Peaks"]

    epochs_start, epochs_end = _ecg_segment_window(rpeaks=rpeaks,
sampling_rate=sampling_rate,
                                                   desired_length=len(ecg_cleaned))
    heartbeats = epochs_create(ecg_cleaned, rpeaks, sampling_rate=sampling_rate,
                               epochs_start=epochs_start, epochs_end=epochs_end)
    return heartbeats

def get_combined_beat_image(signal=None, voltage_range=[-3,3],folder_name=None,
img_name=None):
    beats = ecg_segment(signal, rpeaks=None, sampling_rate=128)
    fig = plt.figure(num=1, figsize=(5,8), dpi=300)
    ax = fig.add_subplot(111)
    ax.axis("off")
    for key in beats.keys():
        ax.plot(beats[key]["Signal"], color="tab:blue", linewidth=1)
    ax.set_ylim(voltage_range)
    fig.tight_layout()
    if not os.path.exists(folder_name):
        os.mkdir(folder_name)
    fig.savefig(os.path.join(folder_name, img_name + ".jpg"))
```

```
    ax.clear()
    fig.clf()

# Creating overlapped beat images for AF
for j in range(0,len(files_list)):
    signal_from_record=pd.read_csv(files_list[j])
    print(files_list[j])
    list_for_signalWithAF=[]
    for i in range(len(signal_from_record)):
        if signal_from_record.loc[i,'PAC_Percent']==0 and
signal_from_record.loc[i,'PVC_Percent']==0 :
            signalWithAF=signal_from_record.loc[i,'0':'3839'].values.tolist()
            list_for_signalWithAF.append(signalWithAF)
    print(len(list_for_signalWithAF))

    if list_for_signalWithAF:
        cleaned_signal=[]
        for k in range(len(list_for_signalWithAF)):
            clean_data=nk.ecg_clean(ecg_signal=list_for_signalWithAF[k],sampling_rate=128)
            cleaned_signal.append(clean_data)
        print(len(cleaned_signal))

        for l in range(len(cleaned_signal)):
            get_combined_beat_image(cleaned_signal[l], folder_name="AF Images for Record "+
str(record_names[j]), img_name= str(1+l))

# Creating overlapped beat images for NSR
for j in range(0,len(files_list_withoutAF)):
    signal_from_record=pd.read_csv(files_list_withoutAF[j])
    print(files_list_withoutAF[j])
    list_for_signalWithoutAF=[]
    for i in range(len(signal_from_record)):
        if signal_from_record.loc[i,'PAC_Percent']==0 and
signal_from_record.loc[i,'PVC_Percent']==0 :
            signalWithoutAF=signal_from_record.loc[i,'0':'3839'].values.tolist()
            list_for_signalWithoutAF.append(signalWithoutAF)
    print(len(list_for_signalWithoutAF))
    if list_for_signalWithoutAF:
        cleaned_signal=[]
        for k in range(len(list_for_signalWithoutAF)):
            clean_data=nk.ecg_clean(ecg_signal=list_for_signalWithoutAF[k],sampling_rate=128)
            cleaned_signal.append(clean_data)
        print(len(cleaned_signal))
        for l in range(len(cleaned_signal)):
            get_combined_beat_image(cleaned_signal[l], folder_name="Record "+
str(record_names_withoutAF[j])+" NSR", img_name= str(1+l))
```

APPENDIX C

TRAINING AND TESTING OF LIGHT WEIGHT NEURAL NETWORKS

```
import os
import cv2
import json
import pathlib
import numpy as np
import pandas as pd
import tensorflow as tf
import albumentations as A
import matplotlib.pyplot as plt
from PIL import Image
from datetime import datetime as dt

from tensorflow import keras
from tensorflow.keras import Model
from tensorflow.keras import layers
from tensorflow.keras.applications import mobilenet_v3
from tensorflow.keras.layers import Dense,GlobalMaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.python.client import device_lib
tf.test.is_built_with_cuda()
print("Num GPUs Available: ", len(device_lib.list_local_devices()))

IMG_SIZE = 224
batch_size=500

%%time
ds_train= tf.keras.preprocessing.image_dataset_from_directory( "D:\\ECG DB\\ECG Beat
Images from LTAF DB", labels='inferred', label_mode='int',
class_names=['AF','NSR','PAC','PVC '], color_mode='rgb', batch_size=batch_size,
image_size=(IMG_SIZE,IMG_SIZE), #reshapeauto
shuffle=True,seed=123, validation_split=0.4, subset="training" )

%%time
ds_validate=tf.keras.preprocessing.image_dataset_from_directory(
   "D:\\ECG DB\\ECG Beat Images from LTAF DB", labels='inferred',label_mode='int',
   class_names=['AF','NSR','PAC','PVC'], color_mode='rgb', batch_size=batch_size,
   image_size=(IMG_SIZE,IMG_SIZE), #reshapeauto
   shuffle=True, seed=123, validation_split=0.4, subset="validation",
 )
```

```
%%time
val_batches = tf.data.experimental.cardinality(ds_validate)
test_dataset = ds_validate.take(val_batches //3)
validation_dataset = ds_validate.skip(val_batches //3)

#Enhance the input pipeline performance while training
%%time
AUTOTUNE = tf.data.AUTOTUNE
train_dataset = ds_train.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)

#Instantiate the base model pretrained on Image Net
%%time
from tensorflow.keras.applications import MobileNetV3Small
model = MobileNetV3Small(input_shape=(224,224,3), weights="imagenet",
include_top=False)

#Add data augmentation into the input pipeline
data_augmentation = tf.keras.Sequential([
tf.keras.layers.experimental.preprocessing.RandomZoom(.5,.5)])

preprocess_input = tf.keras.applications.mobilenet_v3.preprocess_input
rescale = tf.keras.layers.Rescaling(1./223.5, offset=-1)

base_model = tf.keras.applications.MobileNetV3Small(input_shape=(224,224,3),
include_top=False,
                                    weights='imagenet', classifier_activation='softmax')

image_batch, label_batch = next(iter(train_dataset))
feature_batch = base_model(image_batch)
print(feature_batch.shape)
base_model.trainable = False #Freeze the convolutional base

#Customize the classification head of the network
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)
prediction_layer = tf.keras.layers.Dense(1)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)



#Compile the model
inputs = tf.keras.Input(shape=(224,224, 3))
```

```
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = layers.BatchNormalization()(x)
x = layers.Dropout(0.2)(x)
outputs =prediction_layer(x)
mv3small = tf.keras.Model(inputs, outputs)
base_learning_rate = 0.01
mv3small.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
          loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
          metrics=['accuracy'])

#Train and validate the model
%%time
initial_epochs = 20
loss_two, accuracy_two = mv3small.evaluate(validation_dataset)
first_hist = mv3small.fit(train_dataset, epochs=50, validation_data=validation_dataset)

#Plot the learning curve of the model
acc = first_hist.history['accuracy']
val_acc = first_hist.history['val_accuracy']
loss = first_hist.history['loss']
val_loss = first_hist.history['val_loss']

plt.figure(figsize=(8, 8))
plt.style.use('_classic_test_patch')
plt.grid(True)

plt.subplot(2, 1, 1)
plt.style.use('_classic_test_patch')
plt.grid(True)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.style.use('_classic_test_patch')
plt.grid(True)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
```

```
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()


# Confusion Matrix
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn import metrics
y_true=label_batch
y_pred=predictions.NumPy()[0:500]
print(y_pred)
def confusion_metrics (conf_matrix):
    TP = conf_matrix[1][1]
    TN = conf_matrix[0][0]
    FP = conf_matrix[0][1]
    FN = conf_matrix[1][0]

    accuracy = (float (TP+TN) / float(TP + TN + FP + FN))
    misclassification = 1- accuracy
    sensitivity = (TP / float(TP + FN))
    specificity = (TN / float(TN + FP))

    print('-'*50)
    print(f'Accuracy: {(accuracy)}')
    print(f'Sensitivity: {(sensitivity)}')
    print(f'Specificity: {(specificity)}')

confusion_matrix= metrics.confusion_matrix(y_true, y_pred)
# Assigning columns names
confusion_matrix_df= pd.DataFrame(confusion_matrix,
          columns = [ 'Predicted Negative','Predicted Positive'],
          index = [ 'Actual Negative','Actual Positive'])
confusion_matrix_df
```

APPENDIX D

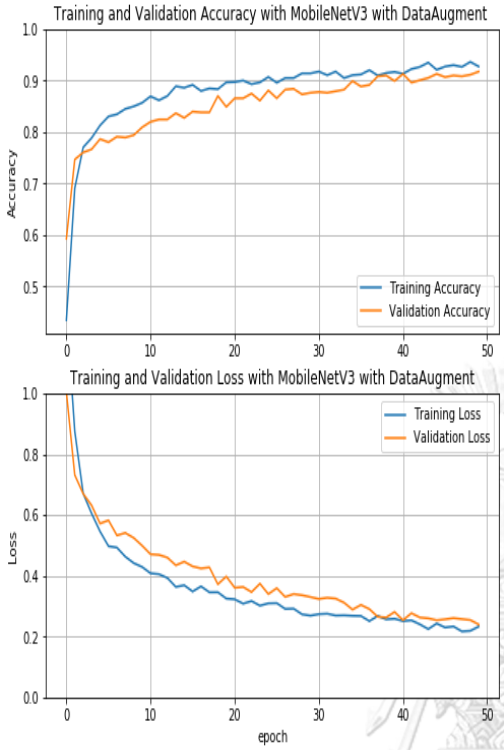


Fig 1 Learning Curve of MobileNetV3 with data augmentation during 50 epochs

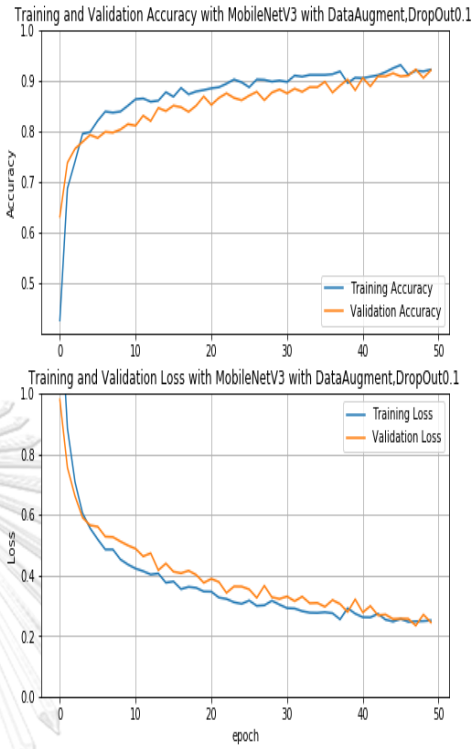


Fig 2 Learning Curve of MobileNetV3 with data augmentation, Dropout value 0.1 during 50 epochs

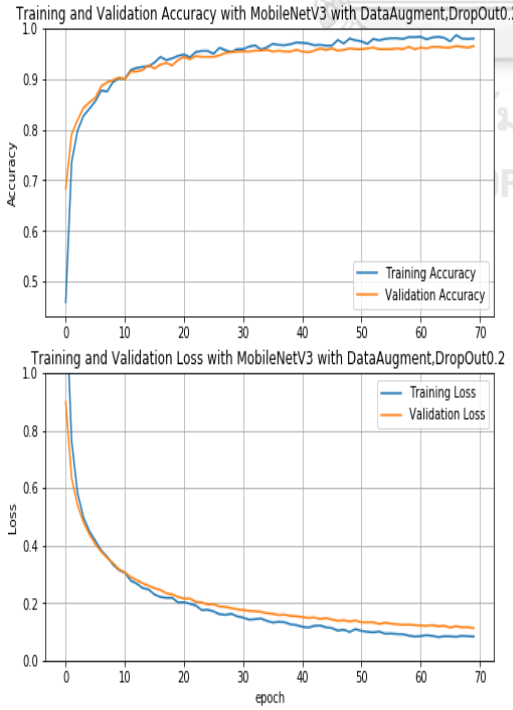


Fig 3 Learning Curve of MobileNetV3 with Data Augmentation and Dropout 0.2 during 70 epochs

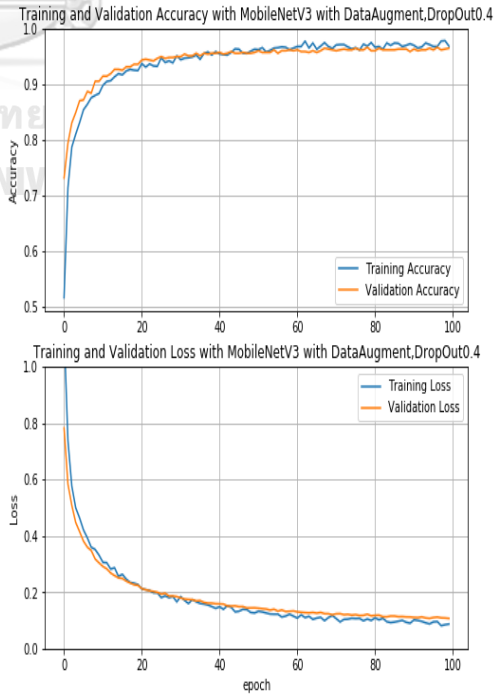


Fig 4 Learning Curve of MobileNetV3 with Data Augmentation, Dropout 0.4 during 100 epochs

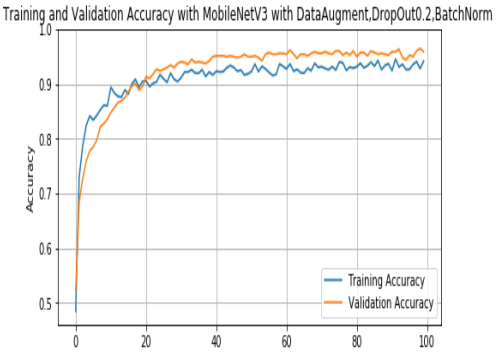


Fig. 5 Learning Curve of MobileNetV3 with Data Augment, Dropout 0.2 and Batch Normalization during 100epochs

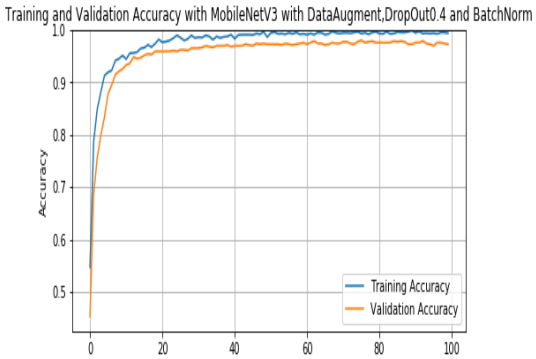


Fig 6 Learning Curve of MobileNetV3 with Data Augment, Dropout 0.4 and Batch Normalization during 100epochs

Table 1 : Original Table from Physionet.org for LTAF Database

Record	Ectopic Beat					Rhythm								
	N	V	A	Q	MISSB	NSR	SVTA	VT	AFIB	B	T	IVR	AB	SBR
0	105416	778	53	-	1	5 (18:31:31)	-	40 (0:42)	44 (2:14:54)	-	-	-	-	-
1	84957	607	4982	-	30	457 (3:38:53)	25 (0:58)	-	53 (16:13:06)	16 (1:46)	6 (0:53)	-	293 (27:36)	83 (10:02)
3	78648	96	2213	-	2	1648 (14:56:49)	11 (0:32)	2 (0:03)	22 (1:20:03)	2 (0:13)	1 (0:14)	-	34 (4:53)	1587 (7:52:53)
5	110925	4	1993	7	7	14 (24:17:02)	2 (0:05)	-	3 (0:40)	-	-	-	8 (0:29)	-
6	100153	5	4505	-	181	48 (24:22:27)	16 (0:29)	-	19 (45:04)	-	-	-	11 (0:44)	1 (0:01)
7	100679	6741	1199	-	3	499 (21:25:26)	28 (1:21)	-	8 (3:24:47)	416 (33:44)	1 (0:07)	-	51 (4:27)	-
8	106872	272	1182	1	1	20 (25:42:48)	13 (0:17)	-	4 (1:38)	1 (0:04)	-	-	1 (0:04)	-
10	105519	279	714	-	1	148 (8:23:01)	68 (3:35)	-	80 (17:07:44)	-	-	-	1 (0:05)	-
11	102489	107	-	-	2	-	-	1 (0:02)	2 (26:03:09)	-	-	-	-	-

12	116650	418	-	75	1	-	-	-	1 (24:05:39)	-	-	-	-	-
13	68236	235	498	-	140	193 (10:08:00)	12 (0:34)	-	2 (4:56:14)	3 (0:20)	1 (0:13)	-	12 (1:56)	172 (4:48:40)
15	85087	199	68	-	306	4 (0:16)	-	-	801 (23:17:57)	-	-	-	-	802 (1:11:39)
16	121308	1876	693	3	1	117 (23:14:58)	4 (0:17)	-	37 (19:57)	7 (1:39)	16 (1:19)	-	39 (8:15)	64 (12:31)
17	138027	32	-	-	1	-	-	-	1 (24:55:10)	-	-	-	-	-
18	141470	42	-	-	1	-	-	-	1 (24:59:16)	-	-	-	-	-
19	104439	4	3455	-	1	83 (23:49:19)	21 (0:52)	-	9 (5:03)	-	-	-	46 (4:05)	6 (0:23)
20	107983	12	-	-	1	-	-	-	2 (24:19:08)	-	-	-	-	1 (0:03)
21	79790	1810	-	-	577	-	-	-	1 (20:56:25)	-	-	-	-	-
22	95430	16651	12677	-	17	1636 (22:37:36)	36 (1:35)	85 (1:55)	102 (23:58)	996 (1:18:11)	282 (27:28)	-	188 (21:01)	107 (8:47)
23	81870	2407	15677	-	3	1194 (16:22:18)	184 (8:00)	1 (0:02)	57 (5:59:42)	172 (34:16)	6 (0:49)	-	801 (2:22:22)	122 (10:25)
24	94465	18	1356	-	17	784 (21:58:46)	9 (0:26)	-	2 (1:47)	-	-	-	16 (2:33)	759 (1:53:36)
25	104065	5899	-	-	1	-	-	212 (7:11)	346 (18:01:49)	-	-	133 (7:19)	-	-
26	68396	200	341	-	9	93 (1:02:02)	4 (0:13)	-	177 (16:43:49)	-	9 (1:18)	-	1 (0:06)	196 (25:25)
28	96525	1046	428	-	1	271 (15:43:05)	29 (1:14)	-	106 (4:43:36)	11 (2:26)	1 (0:11)	-	15 (2:39)	161 (19:30)
30	29389	42	1730	-	1	14 (6:06:41)	8 (0:14)	-	-	-	-	-	5 (0:26)	-
32	86495	149	2551	-	2	541 (19:28:40)	28 (1:14)	2 (0:04)	361 (36:26)	-	-	-	174 (17:58)	-
33	113231	7923	-	-	1	-	-	23 (0:38)	120 (24:09:47)	55 (4:29)	41 (9:31)	-	-	-
34	132424	2742	-	-	1	-	-	3 (0:06)	5 (24:32:53)	-	1 (0:07)	-	-	-
35	88585	19	254	-	13	147 (19:11:21)	5 (0:18)	-	7 (3:52:13)	-	-	-	1 (0:12)	135 (1:17:08)
37	61606	475	2562	-	1	1447 (15:18:23)	38 (3:16)	-	4 (1:29)	-	-	-	56 (6:30)	1392 (4:25:20)
38	80896	129	303	-	2	1236 (18:28:33)	3 (0:10)	-	1 (25:41)	1 (0:05)	-	2 (0:16)	-	1232 (5:27:51)
39	110259	23	3647	-	4	78 (3:21:11)	6 (0:50)	-	141 (21:26:04)	-	-	-	28 (2:59)	56 (8:12)
42	111901	192	2103	-	1	56 (15:01:09)	39 (11:32)	3 (0:04)	122 (5:38:42)	-	-	-	30 (2:32)	-
43	104462	2567	-	-	1	-	-	2 (0:03)	3 (25:35:32)	-	-	-	-	-
44	143	38	-	-	7	-	-	3	4	-	-	-	-	-

	643	03						(0:04)	(25:22:22)						
45	951	30	17	2	1	277	5	-	82	3	-	-	1	191
	13	6	3			(22:35:48)	(0:11)		(2:41:01)	(0:38)			(0:04)	(27:16)
47	834	34	53	-	2	882	18	9		5	8	-	437	417
	44	4	70			(21:25:50)	(6:11)	(0:12)	4 (1:05)	(0:34)	(1:30)		(1:04:41)	(1:49:24)
48	146	64	-	-	1	-	-	12	13	-	-	-	-	-
	116	5						(0:13)	(23:59:23)					
49	794	29	-	-	183	-	-	5	7	-	-	1	-	-
	15	3						(0:20)	(23:58:38)			(0:05)		
51	869	52	20	-	184	348	5	-	68	39	14	-	35	186
	67	47	27			(22:22:04)	(0:10)		(1:26:53)	(3:16)	(1:28)		(3:13)	(15:33)
53	468	8	68	-	747	339	1	-	448	-	-	-	25	714
	03		4			(52:49)	(0:02)		(2:58:48)				(4:53)	(15:43:11)
54	118	11	-	-	1	-	-	-	1	-	-	-	-	-
	187								(24:59:03)					
55	105	26	13	-	2	22	4	-	12	-	-	-	5	-
	478		62			(21:33:46)	(0:11)		(4:02:41)				(0:28)	
56	131	20	13	-	1	4	-	1	1	-	-	-	1	-
	784	8	8			(15:18:16)		(0:02)	(8:46:36)				(0:06)	
58	927	86	14	-	1	928	153	-	12	3	7	-	741	22
	85	3	590			(14:59:38)	(49:00)		(6:16:52)	(0:11)	(0:52)		(1:48:20)	(1:43)
60	116	31	-	-	1	-	-	1	2	-	-	-	-	-
	503	2						(0:02)	(22:19:51)					
62	119	40	-	-	135	-	-	8	41	-	-	-	-	32
	393	13						(0:10)	(24:47:28)					(2:02)
64	119	31	-	-	1	-	-	1	1	-	-	-	-	-
	466	2						(0:02)	(5:42:33)					
65	959	19	-	-	1	-	-	1	2	-	-	-	-	-
	03	3						(0:02)	(25:44:39)					
68	183	11	-	-	1	-	-	5	6	-	-	-	-	-
	691	06						(0:06)	(23:55:35)					
69	140	99	-	-	1	-	-	-	1	-	-	-	-	-
	526	7							(23:40:11)					
70	127	69	-	-	1	-	-	-	1	-	-	-	-	-
	638	7							(26:05:15)					
71	124	41	-	-	1	-	-	-	1	-	-	-	-	-
	789	2							(24:02:05)					
72	141	19	38	-	40	72	5	1	29	41	5	-	2	4 (0:39)
	864	47	7			(3:56:45)	(0:13)	(0:02)	(20:08:31)	(8:25)	(0:46)		(0:52)	
74	886	30	14	-	122	11	-	2	1044	-	-	-	1	1043
	34	04	6		6	(0:29)		(0:03)	(23:32:16)				(0:09)	(1:44:37)
75	126	51	-	-	1	-	-	-	1	-	-	-	-	-
	253	8							(20:48:17)					
100	826	27	60	-	1	839	21	1	659	98	24	-	72	4 (0:19)
	14	23	90			(21:35:34)	(0:45)	(0:03)	(1:53:30)	(9:41)	(2:54)		(5:45)	
101	903	29	48	-	1	675	221	-	147	153	44	-	122	5 (0:41)
	19	96	50			(19:47:36)	(8:56)		(3:20:54)	(16:09)	(5:52)		(18:21)	
102	114	38	24	-	1	46	2	-	43	-	-	-	-	-
	102		3			(23:07:36)	(0:03)		(51:37)					
103	127	21	22	-	1	2095	1369	1	1	8	2	-	824	-
	444	66	04			(9:37:09	(53:47	(0:02	(12:26:5	(0:30)	(0:10)		(1:00:4	

			6			)	)	)	5)				5)	
104	83550	3	601	-	2	29 (17:51:43)	10 (0:18)	-	16 (24:01)	-	-	-	3 (0:11)	-
105	68883	45	70	1	379	398 (13:51:55)	1 (0:03)	-	96 (5:11:07)	-	-	-	1 (0:11)	301 (1:42:58)
110	101827	13	3453	-	1	84 (22:26:20)	1 (0:02)	-	22 (1:09:57)	-	-	-	60 (11:54)	-
111	100583	44	1883	-	2	52 (15:19:46)	15 (0:43)	-	26 (8:26:24)	-	-	-	21 (8:37)	-
112	115263	24	874	-	1	1055 (13:07:24)	-	-	1042 (10:49:36)	-	-	-	7 (0:40)	22 (1:20)
113	112549	28	199	-	1	19 (21:39:33)	-	-	2 (1:58:10)	-	-	-	16 (1:12)	-
114	110181	11	2077	-	1	132 (23:35:38)	35 (1:02)	-	30 (12:47)	-	-	-	76 (10:05)	-
115	108021	28	920	-	2	214 (3:25:19)	19 (0:47)	-	175 (20:15:32)	-	-	-	41 (4:50)	2 (0:07)
116	99617	7	1213 6	-	2	752 (22:29:29)	649 (32:37)	-	25 (24:09)	-	-	-	86 (11:37)	-
117	99533	527	12	-	7	4 (14:47:19)	-	1 (0:01)	5 (9:11:35)	-	-	-	-	-
118	124817	119	216	-	7	7 (19:34:24)	5 (0:10)	1 (0:02)	1 (4:11:33)	-	-	-	-	-
119	96097	956	1701	-	1	340 (20:50:43)	6 (0:16)	37 (0:50)	87 (2:16:45)	-	2 (0:13)	-	52 (6:45)	238 (28:50)
120	109043	14	1382	-	1	53 (23:31:17)	15 (0:42)	-	37 (17:33)	-	-	-	2 (0:11)	-
121	69009	5778	2952	-	212	1335 (11:58:04)	116 (5:19)	2 (0:03)	111 (3:14:08)	298 (49:04)	124 (21:55)	-	23 (3:19)	748 (4:06:28)
122	78127	10016	564	-	3	1078 (13:39:05)	3 (0:05)	-	16 (8:43:01)	368 (34:33)	188 (24:33)	-	7 (0:32)	514 (36:47)
200	77769	4846	2	-	1	11 (1:14)	-	5 (0:09)	19 (23:50:26)	-	2 (0:15)	-	-	-
201	85934	89	-	-	1443	-	-	-	7 (23:58:08)	-	-	-	-	6 (0:31)
202	141501	4122	-	-	1	-	-	-	1 (23:59:51)	-	-	-	-	-
203	73198	641	-	-	1	-	-	4 (0:19)	6 (18:09:04)	-	-	1 (0:03)	-	-
204	118880	17240	-	-	1	-	-	333 (9:56)	334 (20:35:58)	-	-	-	-	-
205	116473	27	-	-	1	-	-	-	1 (23:49:35)	-	-	-	-	-
206	121059	351	-	-	2	-	-	4 (0:05)	6 (20:47:16)	-	-	-	-	1 (0:03)
207	130780	479	-	-	3	-	-	16 (0:42)	17 (23:49:22)	-	-	-	-	-
208	120658	84	-	-	8	-	-	-	1 (23:55:16)	-	-	-	-	-

REFERENCES

[1] N. T. Srinivasan and R. J. Schilling, "Sudden cardiac death and arrhythmias," *Arrhythmia & electrophysiology review,* vol. 7, no. 2, p. 111, 2018.
[2] M. Nagashima *et al.*, "Cardiac arrhythmias in healthy children revealed by 24-hour ambulatory ECG monitoring," *Pediatric cardiology,* vol. 8, pp. 103-108, 1987.
[3] S. K. Kunutsor and J. A. Laukkanen, "Cardiovascular complications in COVID-19: a systematic review and meta-analysis," *Journal of Infection,* vol. 81, no. 2, pp. e139-e141, 2020.
[4] X. Li *et al.*, "Cardiac injury associated with severe disease or ICU admission and death in hospitalized patients with COVID-19: a meta-analysis and systematic review," *Critical care,* vol. 24, pp. 1-16, 2020.
[5] A. S. Manolis, A. A. Manolis, T. A. Manolis, E. J. Apostolopoulos, D. Papatheou, and H. Melita, "COVID-19 infection and cardiac arrhythmias," *Trends in cardiovascular medicine,* vol. 30, no. 8, pp. 451-460, 2020.
[6] C.-E. Chiang, K.-L. Wang, and G. Y. Lip, "Stroke prevention in atrial fibrillation: an Asian perspective," *Thrombosis and haemostasis,* vol. 112, no. 05, pp. 789-797, 2014.
[7] G. D. Clifford, F. Azuaje, and P. Mcsharry, "ECG statistics, noise, artifacts, and missing data," *Advanced methods and tools for ECG data analysis,* vol. 6, no. 1, p. 18, 2006.
[8] M. M. Hadhoud, M. I. Eladawy, and A. Farag, "Computer aided diagnosis of cardiac arrhythmias," in *2006 International Conference on Computer Engineering and Systems*, 2006: IEEE, pp. 262-265.
[9] A. Shiyovich, A. Wolak, L. Yacobovich, A. Grosbard, and A. Katz, "Accuracy of diagnosing atrial flutter and atrial fibrillation from a surface electrocardiogram by hospital physicians: analysis of data from internal medicine departments," *The American journal of the medical sciences,* vol. 340, no. 4, pp. 271-275, 2010.
[10] S. M. Mathews, C. Kambhamettu, and K. E. Barner, "A novel application of deep learning for single-lead ECG classification," *Computers in biology and medicine,* vol. 99, pp. 53-62, 2018.
[11] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ECG classification by 1-D convolutional neural networks," *IEEE Transactions on Biomedical Engineering,* vol. 63, no. 3, pp. 664-675, 2015.
[12] G. Sannino and G. De Pietro, "A deep learning approach for ECG-based heartbeat classification for arrhythmia detection," *Future Generation Computer Systems,* vol. 86, pp. 446-455, 2018.
[13] A. Howard *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314-1324.
[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510-4520.
[15] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848-6856.

[16] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116-131.

[17] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019: PMLR, pp. 6105-6114.

[18] Y. Zhou, S. Chen, Y. Wang, and W. Huan, "Review of research on lightweight convolutional neural networks," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020: IEEE, pp. 1713-1720.

[19] M. Abdelazez, S. Rajan, and A. D. Chan, "Transfer learning for detection of atrial fibrillation in deterministic compressive sensed ECG," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2020: IEEE, pp. 5398-5401.

[20] J. Chanthercrob, S. Mahattanatawee, A. Teeramongkonrasmee, and P. Somboon, "Development of rhythm-based and morphology-based algorithm for atrial fibrillation detection from single lead ecg signal," in *2020 8th International Electrical Engineering Congress (iEECON)*, 2020: IEEE, pp. 1-4.

[21] Y. Liu *et al.*, "Precise and efficient heartbeat classification using a novel lightweight-modified method," *Biomedical Signal Processing and Control,* vol. 68, p. 102771, 2021.

[22] J. S. Steinberg, H. O'Connell, S. Li, and P. D. Ziegler, "Thirty-second gold standard definition of atrial fibrillation and its relationship with subsequent arrhythmia patterns: analysis of a large prospective device database," *Circulation: Arrhythmia and Electrophysiology,* vol. 11, no. 7, p. e006274, 2018.

[23] T. J. Jensen, J. Haarbo, S. M. Pehrson, and B. Thomsen, "Impact of premature atrial contractions in atrial fibrillation," *Pacing and clinical electrophysiology,* vol. 27, no. 4, pp. 447-452, 2004.

[24] Y. G. Kim *et al.*, "Premature ventricular contraction is associated with increased risk of atrial fibrillation: a nationwide population-based study," *Scientific reports,* vol. 11, no. 1, p. 1601, 2021.

[25] G. M. Marcus and T. A. Dewland, "Premature atrial contractions: a wolf in sheep's clothing?," vol. 66, ed: American College of Cardiology Foundation Washington, DC, 2015, pp. 242-244.

[26] J. Burnett, "The origins of the electrocardiograph as a clinical instrument," *Medical History,* vol. 29, no. S5, pp. 53-76, 1985.

[27] W. B. Fye, "A history of the origin, evolution, and impact of electrocardiography," *The American journal of cardiology,* vol. 73, no. 13, pp. 937-949, 1994.

[28] A. Bansal and R. Joshi, "Portable out-of-hospital electrocardiography: A review of current technologies," *Journal of arrhythmia,* vol. 34, no. 2, pp. 129-138, 2017.

[29] P. A. Iaizzo, *Handbook of cardiac anatomy, physiology, and devices*. Springer Science & Business Media, 2010.

[30] S. Vieau and P. A. Iaizzo, "Basic ECG theory, 12-lead recordings, and their interpretation," *Handbook of Cardiac Anatomy, Physiology, and Devices,* pp. 321-334, 2015.
[31] N. Lowres *et al.*, "Feasibility and cost-effectiveness of stroke prevention through community screening for atrial fibrillation using iPhone ECG in pharmacies," *Thrombosis and haemostasis,* vol. 111, no. 06, pp. 1167-1176, 2014.
[32] K. Kearley *et al.*, "Triage tests for identifying atrial fibrillation in primary care: a diagnostic accuracy study comparing single-lead ECG and modified BP monitors," *BMJ open,* vol. 4, no. 5, p. e004565, 2014.
[33] P. J. Podrid and P. R. Kowey, *Cardiac arrhythmia: mechanisms, diagnosis, and management*. Lippincott Williams & Wilkins, 2001.
[34] G. Tse, "Mechanisms of cardiac arrhythmias," *Journal of arrhythmia,* vol. 32, no. 2, pp. 75-81, 2016.
[35] F. Kaasenbrood *et al.*, "Opportunistic screening versus usual care for diagnosing atrial fibrillation in general practice: a cluster randomised controlled trial," *British Journal of General Practice,* vol. 70, no. 695, pp. e427-e433, 2020.
[36] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A review on deep learning methods for ECG arrhythmia classification," *Expert Systems with Applications: X,* vol. 7, p. 100033, 2020.
[37] D. Gupta, B. Bajpai, G. Dhiman, M. Soni, S. Gomathi, and D. Mane, "WITHDRAWN: Review of ECG arrhythmia classification using deep neural network," ed: Elsevier, 2021.
[38] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, and T. S. Huang, "Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7268-7277.
[39] L. Zhou, C. Zhang, and M. Wu, "D-LinkNet: LinkNet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 182-186.
[40] J. Dai *et al.*, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764-773.
[41] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels," *arXiv preprint arXiv:1907.09595,* 2019.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.
[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015: PMLR, pp. 448-456.
[44] P. Shimpi, S. Shah, M. Shroff, and A. Godbole, "A machine learning approach for the classification of cardiac arrhythmia," in *2017 international conference on computing methodologies and communication (ICCMC)*, 2017: IEEE, pp. 603-607.
[45] P.-Y. Hsu and C.-K. Cheng, "Arrhythmia classification using deep learning and machine learning with features extracted from waveform-based signal

processing," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2020: IEEE, pp. 292-295.

[46] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical systems and signal processing,* vol. 151, p. 107398, 2021.

[47] M. Salem, S. Taheri, and J. S. Yuan, "ECG arrhythmia classification using transfer learning from 2-dimensional deep CNN features," in *2018 IEEE biomedical circuits and systems conference (BioCAS)*, 2018: Ieee, pp. 1-4.

[48] B. M. Mathunjwa, Y.-T. Lin, C.-H. Lin, M. F. Abbod, and J.-S. Shieh, "ECG arrhythmia classification by using a recurrence plot and convolutional neural network," *Biomedical Signal Processing and Control,* vol. 64, p. 102262, 2021.

[49] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *circulation,* vol. 101, no. 23, pp. e215-e220, 2000.

[50] T.-C. Huang, P.-T. Lee, M.-S. Huang, P.-F. Su, and P.-Y. Liu, "Higher premature atrial complex burden from the Holter examination predicts poor cardiovascular outcome," *Scientific reports,* vol. 11, no. 1, p. 12198, 2021.

[51] D. Makowski *et al.*, "NeuroKit2: A Python toolbox for neurophysiological signal processing," *Behavior research methods,* pp. 1-8, 2021.

[52] Y. G. Kim *et al.*, "Premature ventricular contraction is associated with increased risk of atrial fibrillation: a nationwide population-based study," *Scientific reports,* vol. 11, no. 1, pp. 1-8, 2021.

VITA

NAME	Miss Khaing Su Thway
DATE OF BIRTH	13 March 1995
PLACE OF BIRTH	Mandalay, Myanmar
INSTITUTIONS ATTENDED	Chulalongkorn University
HOME ADDRESS	Room 407, Building No. 132, ABLE26 Apartment, Soi Phahoyothin 26, Yak 5, Chom Phom, Chatuchuk, Bangkok
PUBLICATION	https://ieeexplore.ieee.org/abstract/document/9894861
AWARD RECEIVED	Graduate_Scholars for ASEAN and Non-ASEANs scholarship program