Master Thesis

Combining multi-scale Kernels and Transformer Encoder for multi-label ECG Classification

Kilian Laurin Kramer

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science of Artificial Intelligence
at the Department of Advanced Computing Sciences
of the Maastricht University

Thesis Committee:

Dr. Pietro Bonizzi
Dr. Stef Zeemering
Dr. Joël Karel

Maastricht University
Faculty of Science and Engineering
Department of Advanced Computing Sciences

August 30, 2024

Contents

Abstract

The accurate classification of electrocardiogram (ECG) signals is crucial for diagnosing various cardiovascular diseases, such as Atrial Fibrillation, and to provide cardiologists with reliable ECG arrhythmia classification tools. With recent advances in deep-learning, Transformer models have emerged as powerful models for the accurate single- and multi-lead ECG classification of Atrial Fibrillation. This thesis investigates the potential of applying Transformer models for the complex task of classifying into 26 different cardiac arrhythmias from single- and multi-lead ECGs. The work proposes a deep residual multi-scale Convolutional Network combined with several Transformer encoder blocks, which shows improved performances to a conventional Transformer encoder model trained on raw ECG data. By incorporating a state-of-the-art custom loss function the model is able to provide competitive results on the Physionet 2021 challenge data. In the experiments a feature-based random forest classifier and several other residual convolutional and attention-based deep-learning models are compared. The residual Convolutional Network and random forest classifier show highest accuracy, while the random forest classifier is much less computationally expensive. Finally, the generalization ability of the residual Convolutional Network is evaluated across different ECGs datasets.

Chapter 1

Introduction

1.1 Problem statement

Cardiovascular diseases, such as Atrial Fibrillation, are among the leading causes of death in the population, resulting in an increased demand for efficient, timely, and reliable cardiac assessment. Deep-learning approaches can be used to develop multi-classification models for arrhythmia detection and improve medical monitoring. With recent advances in deep-learning models, Transformer-based architectures have emerged as powerful tools for accurate single-lead and multi-lead ECG classification [3] [5] [9] [22] [19] [26] [46] [50]. Multi-lead ECGs provide a view of the heart's electrical activity from different angles and can localise abnormalities more reliable. It is a standard in clinical monitoring. Accurate single-lead ECG classification is an attractive area of research when it comes to continuous processing of single-lead ECGs by remote monitoring systems, such as wearable devices like Apple Watches [19], for the continuous monitoring of an individual's condition and to provide early suggestions for a doctoral visit. Today, much research and accurate models are available for simpler arrhythmia classification tasks, e.g. binary AFIB (Atrial Fibrillation) classification or grouped common arrhythmia types (i.e. AAMI standard). The problem, professional treatment requires individual and detailed ECG assessment. Complex multi-label classification tasks is an ongoing research topic and accurate models are still limited on comprehensive arrhythmia detection tasks including Transformer models and rare subdiseases, such as Atrial Flutter, Premature Ventricular Contractions and other. This thesis investigates the potential of applying Transformer-based models to ECG classification. It presents a novel approach that uses a multi-scale Convolutional Network in combination with several Transformer encoder blocks. Furthermore, it provides a comparison with non deep-learning, based on extracted features, and several other (attention-based) deep-learning based models, such as deep residual Convolutional Networks. Parts of the experiments focus in particular on the classification of Sinus Rhythm, Atrial Fibrillation, Atrial Flutter, Premature Atrial Contractions and Premature Ventricular Contractions.

1.1.1 Electrocardiograms

Electrocardiograms (ECGs) monitor the condition of a patient's heart. An ECG measures the electrical signals flowing through the heart in repeated cardiac cycles. ECGs are an essential tool for cardiologists to diagnose heart diseases. To measure the heart's activity electrodes, called leads within the ECG, are placed on the skin of the upper chest and back. The number of leads affects the quality of the measurements, i.e. single-lead ECGs are based on two electrodes, while

multi-lead ECGs use more than two electrodes. Multi-lead ECGs generally include up to 12 leads, defined in the literature as I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5 and V6 [2] [23]. Multi-lead ECGs are used in preference to single-lead ECGs because they provide a view of the heart from different angles and can therefore localise abnormalities more accurate through spatial information. Typically, these recordings last from a few seconds to a few minutes and are called short-term ECGs. However, sometimes long-term ECGs are performed. Long-term ECGs monitor the heart's activity over a longer period of time, e.g. 24 hours, and provide a more complete picture. This is necessary to detect abnormalities that are less common and harder to detect. The scope of this thesis focuses on short-term single-lead ECGs. An electrocardiogram can be recorded with different devices, e.g. in clinical settings 12-lead recordings are standard, while for remote monitoring often portable devices are used, such as the Holter monitoring system. The Holter monitoring system can be adapted to different numbers of leads as required, e.g. three leads or up to 12 leads. Special watches [19] can now also be used to record the heart's activity. These watches use a technology called Photoplethysmography (PPG), where LED lights under the watch emit light into the skin [30]. The light is absorbed and reflected by blood vessels and the changes in light absorption are measured to determine heart rate. However, PPG is a measure of the blood pressure from which the heart rate can be derived, but which is not direct an ECG recording of the electrical activity of the heart. Usually, the derived quality of the sensor is not as accurate as a standard ECG, making it a less reliable method. A watch is still an interesting device because it can monitor a patient's heart condition on a regular basis. It also has the advantage of not requiring the involvement of a cardiologist and can provide early advice for a more transparent in-house check in a doctor's surgery. Accuracy in arrhythmia detection is important for several reasons, including arrhythmia risk and correct treatment. For example, increased accuracy in automated ECG processing can reduce false positives and false negatives. This can avoid unnecessary doctoral visits and lead to more efficient and effective clinical monitoring and treatment. In addition to developing accurate models a goal of this thesis will be to transfer and apply the pre-trained models from the Physionet 2021 challenge data [31], which this thesis utilizes for training all models, to the database provided by the Maastricht University, here referred to as the MyDiagnostick database. This is particularly interesting for assessing the generalisability of the models. Both datasets contain recordings from different monitoring systems. Analyses and pre-processing steps need to take into account the type of monitoring system and electrodes, number of leads, recording duration, sampling rate and post-processed filters applied, which will be discussed in section 4.1.3.

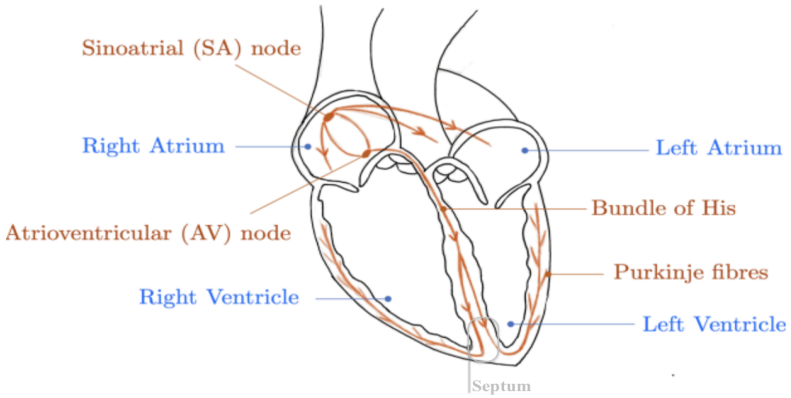

Figure 1.1: Human heart anatomy
Source: Collimator

Figures 1.1 and 1.2 illustrate the activity of a heart from a physiological perspective and a single normal heartbeat from an ECG, defined as a Sinus Rhythm. In the literature [2] [23] a heartbeat is described by the P, Q, R, S, T waves, segments and their intervals. Each wave and segment corresponds to a specific event in the electrical cycle of the heart. In the following a brief description of the entire contraction and depolarisation process of a single Sinus Rhythm heartbeat is provided:

The two graphs show a single heart contraction and depolarisation recorded on an ECG, starting with the electrical impulse from the Sinoatrial (SA) node and the atrial contraction, followed by the depolarisation phase and the pumping of blood from the ventricles into the aorta and pulmonary artery and the final repolarisation and relaxation of the heart. The SA node, located in the right atrium, is the heart's natural pacemaker. It initiates all heartbeats and determines the heart rate. The P wave in an ECG shows the electrical activation of the SA node, which causes the atria of the heart to contract. It is the first wave in a heartbeat and is usually small and positive. When the atria of the heart fill with blood, the SA node fires and the electrical impulse from the SA node spreads to the two upper atria, causing them to contract. Atrial

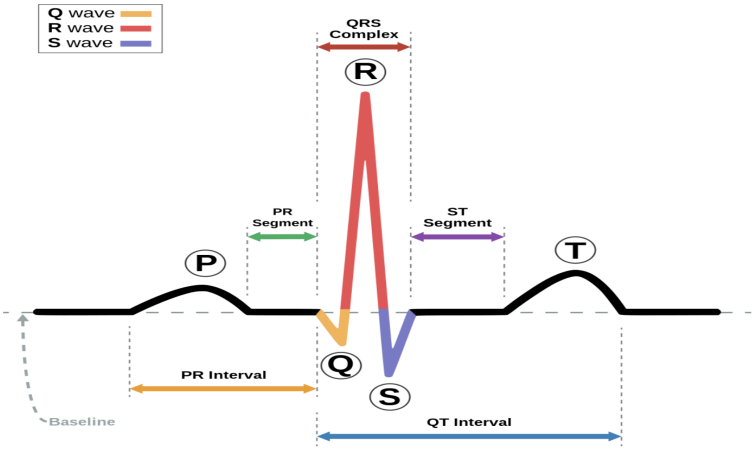

Figure 1.2: Heartbeat rhythm composition
Source: Wikipedia

depolarisation, which fills the blood from the atria into the ventricles, begins in a normal Sinus Rhythm about 100 milliseconds after the SA node fires. The Atrioventricular (AV) node, located opposite the SA node in the right atrium, receives the electrical signal from the SA node and marks the start of ventricular depolarisation. The electrical depolarisation is slowed down before it propagates to the ventricles, here part of the PR interval. The PR segment represents the duration it takes for the electrical signal to travel from the SA node to the ventricles. The AV node acts as an electrical gateway to the ventricles and delays the electrical conduction to the ventricles. This delay ensures that the atria contract all the blood into the ventricles before the ventricles depolarise. The AV node conducts the electrical impulse to the AV bundle, also called the Bundle of His. The Bundle of His is divided into right and left branches, through which the electrical impulses are conducted to the Purkinje Fibres and causes the main depolarisation effect of the ventricles. This ventricular depolarisation is seen on the ECG as the large QRS complex. The Q wave corresponds to the depolarisation of the interventricular septum. The R wave is produced by the main depolarisation of the ventricles. The S wave represents the final

phase of ventricular depolarisation. Atrial repolarisation also occurs during this time, but the signal is obscured by the large QRS complex. The ST segment, which follows the QRS complex, is the initial phase of the ventricular repolarisation. The ST segment reflects when the ventricles contract, pumping oxygen rich blood and oxygen poor blood into the aorta and pulmonary artery. The final T wave represents ventricular repolarisation before the heart relaxes. Repolarisation continues until the end of the T wave. The QT interval represents the entire ventricular contraction activity (depolarisation and repolarisation). At the end of this process, the atria fill with blood again and the SA node fires to repeat the cardiac cycle. This process represents a normal Sinus Rhythm.

1.1.2 Arrhythmia diseases

In the literature [2] [23], normal Sinus Rhythm is described as being within normal limits and ranges. For Sinus Rhythm, this is usually between 60 and 100 beats per minute. A rate below 60 is generally defined as Sinus Bradycardia and a rate above 100 beats per minute as Sinus Tachycardia. The two classes group several arrhythmia subdiseases. Other arrhythmia types are about the heart beating irregularly, sometimes also in combination with Bradycardia or Tachycardia. Arrhythmia subtypes are characterised by different origins and causes in the heart and require individual treatment. In the Physionet 2021 challenge data more than 100 arrhythmia subtypes are present 4.1. For the evaluation of the transferred models from pre-training on the Physionet 2021 challenge data and testing on the MyDiagnostick data, this thesis focuses the classification on Sinus Rhythm (SR), Atrial Fibrillation (AF), Atrial Flutter (AFL), Premature Atrial Contractions (PAC) and Premature Ventricular Contractions (PVC). This is due to the limited class annotation of the MyDiagnostick dataset, which only consists of these class labels. However, a broader performance evaluation of the models is performed on 26 different classes of the annotated Physionet 2021 challenge data. Most of the more than 100 available classes in the Physionet 2021 challenge data contain few examples, so the challenge organisers use a subset of 26 classes to benchmark the models within the official challenge. Much research has been done to develop accurate models for binary classification of non-abnormality vs. abnormality, i.e. Sinus Rhythm vs. Atrial Fibrillation. However, fewer research has been done on distinguishing specific arrhythmia subtypes, such as Atrial Fibrillation and Atrial Flutter. The limited research in this area tends to show weaker outcomes compared to models that classify between Sinus Rhythm and Atrial Fibrillation [33] [32]. Atrial Fibrillation and Atrial Flutter are often confused by algorithms, because they have similar characteristics. PAC and PVC are also examined, because the MyDiagnostick dataset provided contains these annotations. In the following is a brief summary of a normal Sinus Rhythm and each arrhythmia type studied:

- Sinus Rhythm (SR): Sinus Rhythm is characterised by regular heartbeats with a frequency rate between 60 and 100 beats per minute. Each P wave is followed by a narrow QRS complex, that typically lasts between 80 and 100 milliseconds. The PR intervals are constant.

- Atrial Fibrillation (AF): Atrial Fibrillation is an abnormal electrical activity that causes the atrial muscle to contract at different times. Atrial Fibrillation is characterised by a rapid and irregular heartbeat. These uncoordinated contractions produce a quivering or fibrillating activity. Atrial Fibrillation does not have constant P waves preceding the QRS complexes. Although, the fibrillation effect may resemble a P wave at times when it is not expected. Only some of the electrical signals are conducted down into the ventricles, resulting in ventricular depolarisation. There is no real pattern which impulses are conducted. In the literature [2] [23], AF is also described as an irregular heart rhythm, which corresponds to the irregular intervals between the RR intervals.

- Atrial Flutter (AFL): Atrial Flutter is characterised by flutter waves, also known as a "sawtooth" pattern, caused by rapid atrial depolarisation activity. Atrial Flutter occur through coordinated electrical activity in the atria due to a re-entry pathway and a rapid contraction of the atria. This results in around 250 to 300 beats per minute with a regular atrial rate and a narrow QRS complex. However, the AV conducts the signal slower, causing a slowed ventricular depolarization. Therefore, Atrial Flutter is also characterised by the number of P waves compared to the number of ventricular contractions, which shows the ratio of non conducted to conducted beats, e.g. a 3:1 conduction means that every third atrial impulse is conducted to the ventricles. A conduction ratio of 1:1 is also possible and is associated with instability and progression to ventricular fibrillation. These ratios can be variable in the same patient, making the P-waves intervals irregular, why it can be often confused with Atrial Fibrillation [33] [42] [32]. Atrial Flutter is less dangerous than Atrial Fibrillation, but can progress dangerous if left untreated.

- Premature Atrial Contractions (PAC): Premature Atrial Contractions are heartbeats that originate in the atria. Premature Atrial Contractions are characterised by an abnormal P wave morphology compared to normal Sinus Rhythm P waves. On an ECG, PACs occur as premature and extra beats that disrupt the regular heart rhythm. The QRS complex that follows the PAC may resemble a pause. However, if the locations of the P waves are followed they should approximately occur in equal time intervals. Normally, PACs are common finding on ECGs and usually do not require further investigation.

- Premature Ventricular Contractions (PVC): Premature Ventricular Contractions are early heartbeats that originate in the Purkinje Fibre region of the ventricles. They are also identified by extra and abnormal beats. On an ECG, PVCs are seen as wide and bizarre QRS complexes that are not preceded by a P wave. The QRS complex in PVCs is longer than 120 milliseconds and there is a compensatory pause before the next beat. Unless PVCs are not frequent, i.e. if they occur more than 10 to 30 times per hour, they are not dangerous on their own. If they occur every beat in a row, they can be diagnosed as Ventricular Tachycardia, which is critical for stroke.

Figure 1.3 shows for a normal Sinus Rhythm and each of the investigated arrhythmias an example from the Physionet 2021 challenge database.

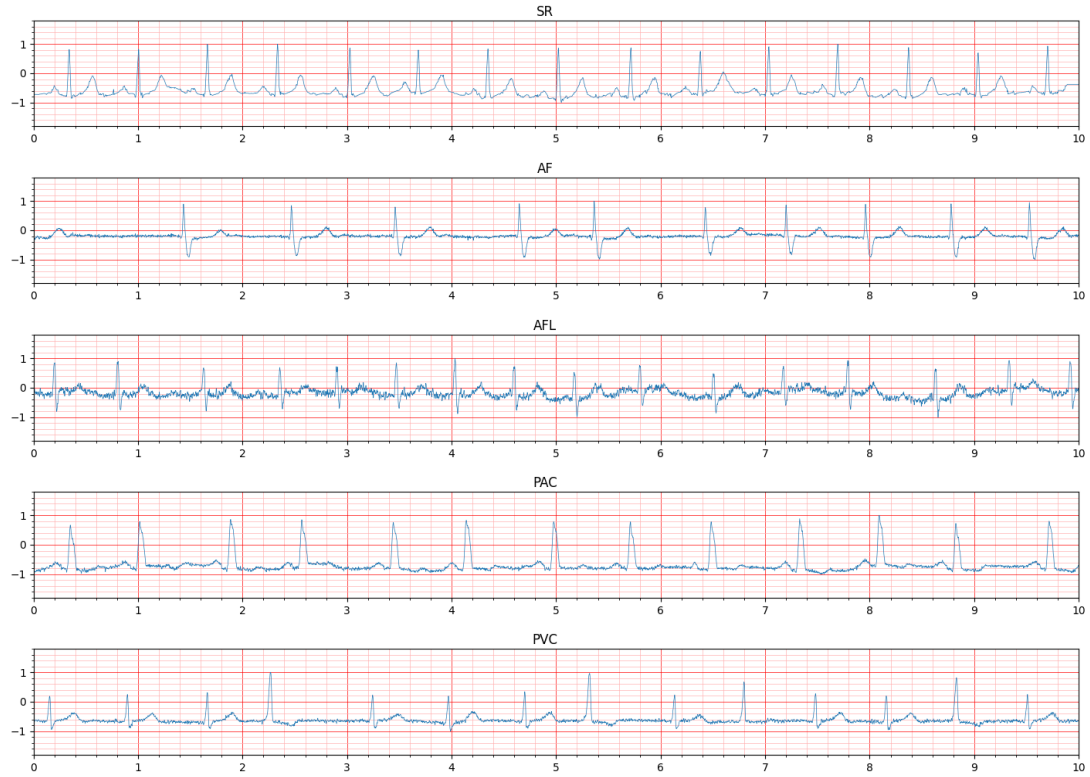

Figure 1.3: Examples for Sinus Rhythm (1), Atrial Fibrillation (2), Atrial Flutter (3), Premature Atrial Contractions (4) and Premature Ventricular Contractions (5) from the Physionet 2021 database

1.2 Goal

The main goal of this thesis is to investigate and compare different Transformer architectures with a non deep-learning model, based on extracted features, and other deep-learning approaches. The main advantage of the Transformer model is the attention mechanism, which is designed to learn relationships within data and attend to specific areas in the input. This could be beneficial for tasks, such as classifying Atrial Fibrillation in ECGs, where relationships within the data could lead to a better understanding of underlying patterns that depend on specific events. A Convolutional Network focuses on local patterns and is less suited for understanding distant relationships within extracted features. This thesis explores the attention mechanism for extracting the harder to detect abnormalities, such as distinguishing between Atrial Fibrillation and Atrial Flutter, or capturing Premature Atrial Contractions and Premature Ventricular Contractions, which often can occur only once on an ECG. The research question is whether the Transformer model with the attention mechanism can capture these patterns better than a traditional Convolutional Network. In addition, the combination of a Convolutional Network and Transformer model, other attention-based mechanisms and various Transformer configurations are investigated and analysed, e.g. signal preparation into segments, applied positional encoding, number of encoder blocks and heads, attention matrices dimensions, feed-forward layer dimension and dropout rate. Objective will also be to reduce the model size, while aiming for high

performance. For the evaluation of the models the Physionet 2021 challenge data [31] and the provided MyDiagnostick databases will be used. First, this thesis will evaluate the models on the Physionet 2021 challenge data itself, and then transfer and evaluate the generalisation ability of the best performing pre-trained model from the Physionet 2021 challenge data on the MyDiagnostick database. Necessary pre-processing steps and considerations are discussed in section 4.1.3. Although, the Physionet 2021 challenge data provides 12-lead ECGs, this thesis will limit the experiments to single-lead ECGs, which is due that the provided MyDiagnostick dataset contains only of single-lead ECGs. Four research questions are formulated, which will be addressed in the experiments 4.2 and answered by the end of this thesis 5.2:

1.2.1 Research questions

1. How well does a Transformer-based model perform on the Physionet 2021 challenge data compared to a feature-based model or a Convolutional Network?

2. Can an ensemble Transformer model and Convolutional Network effectively capture spatio-temporal information and improve accuracy?

3. How is the performance of the most promising deep-learning model at discriminating SR, AF, AFL, PAC and PVC on both datasets?

4. What are the challenges in transferring the pre-trained models from the Physionet 2021 challenge data to the MyDiagnostick database? Do the models generalise well, even though different ECG devices were used?

1.3 Thesis overview

Chapter 1 provided an introduction to the problem statement, domain and the research objective of this thesis. Chapter 2 discusses related work in this area. Chapter 3 explains the relevant mathematical background and presents several developed approaches, as well as the proposed approach 3.3. Chapter 4 discusses the experiments and evaluates the models. Chapter 5 summarises the results, answers the research questions and gives an outlook for further research.

Chapter 2

Related work

This chapter discusses related work. It is divided into three sections. The first section discusses related work on models developed for the Physionet 2021 challenge, which provides an annotated dataset for the classification of 26 different arrhythmia types. The second section discusses related work on Transformer-based models for the general classification of various heart arrhythmias from ECGs. The final section discusses related work in the specific area of classifying Sinus Rhythm, Atrial Fibrillation, Atrial Flutter, Premature Atrial Contraction and detection of Premature Ventricular Contraction based on deep-learning models.

2.1 Related work on the developed models for the Physionet 2021 challenge

Several paper in the Physionet 2021 challenge incorporate an attention-based mechanism. To give an overview, the papers from the four highest ranked models with the best performances within the challenge are discussed, from which the first, second and fourth ranked model utilize an attention-based mechanism. The fourth ranked paper highlights several different attention-based methods and is discussed in more detail. The first ranked model of the Physionet 2021 challenge by Nejedly et al. [27] proposes a deep residual Convolutional Network combined with a single Multi-head Attention layer. As all approaches in the Physionet 2021 challenge, their model is designed for 12-lead ECG classification, while for fewer lead configurations the unused leads are padded with zeros. Their approach achieves about 58% on all 2, 3, 4, 6 and 12-lead tasks within the challenge metric and is able to achieve on all lead tasks the 1st ranked performances. Key features of their model are that the model is based on the ResNet architecture proposed by Zhang et al. [16], which introduced the idea of residual connections to develop a deep Convolutional Network. Nejedly et al. use large convolutional filters, i.e. 15x15 on the first layer and 9x9 in subsequent residual blocks. The model is composed of five residual blocks, where each block consists of two convolutional layers, a batch normalisation layer and a feed-forward layer with a leaky relu activation function. The model concatenates to the residual blocks a single Multi-head Attention block, as the authors justify to also capture temporal relations [27]. However, notable is a follow-up study [28] on the same experiments, where the authors find that the Multi-head Attention layer slightly worse the classification performance of the model. Based on the experiments the authors find that without the Multi-head Attention block the model achieves an overall performance of 58% and with the Multi-head attention block only 57% on the same Physionet 2021 challenge test data. A question that arises from this, is whether the features that are passed

into the attention block are not appropriate prepared. In their paper the authors do not explain this in detail. Furthermore, the authors do not state whether they apply positional encoding, that potentially could help the attention mechanism with extracting relational patterns. Another key feature of their approach is that the authors propose a cost function that incorporates the Physionet 2021 challenge evaluation metrics into the loss to optimize the model on the challenge metric. The cost function is an altered version from Vicar et al. [40] [41]. This thesis will adopt the cost function for fine-tuning the models on the Physionet 2021 data, which is discussed in section fine-tuning 3.3.8. The second ranked paper by Han et al. [15] achieves between 55%-58% in the Physionet 2021 challenge. In their approach the authors use a deep residual Convolutional Network combined with demographic features, i.e. age and gender. The features are directly incorporated into the final output dense layer of the network. Within the model a Squeeze and Excication Network is integrated. A Squeeze and Excication Network is based on a mechanism called channel-attention proposed by Hu et al. in [21]. This thesis benchmarks a Squeeze and Excication Network as it follows a related attention mechanism approach compared to the idea of Transformer models. The main ideas of the Squeeze and Excitation Network will be discussed in section 3.3.5. The authors of [15] highlight that the model achieves high generalisation performance by using a "leave-one-dataset-out-cross-validation" strategy. In addition, their approach use a data augmentation method called "Mixup" [48], which smoothes the decision boundaries of the model through a regularisation technique that mixes two input samples with their features and labels based on a coefficient. The third ranked paper by Wickramasinghe and Athif [44] proposes two combined deep residual Convolutional Networks with four blocks each, which achieves an accuracy of 51%-55% on the final test set. The two networks work in parallel, while one Convolutional Network receives the raw ECG signal with temporal information as input and the second Convolutional Network applies beforehand a Fast Fourier Transformation (FFT) on the ECGs to capture features from the frequency domain. Both models are combined by a dense and pooling layer in the output. The authors apply standard pre-processing steps, such as normalisation, resampling and zero-padding. The fourth ranked paper by Srivastava et al. [37] applies a residual Convolutional Network, which incorporates several convolution- and attention-based methods. The methods include "inception" proposed in the GoogLeNet by Szegedy et al. in [38], channel-attention based on the Squeeze and Excitation Network architecture [21] and attention-pooling proposed by Ilse et al. in [24]. This thesis will also experiment with a modified version of the Inception Network. The main ideas of the Inception Network will be discussed in 3.3.4. Although, the original paper apply 1x1 convolutions in the inception module, the authors from the challenge paper [37] do not apply 1x1 convolutions. Instead they use 1D convolutions with dimensions 1x3, 1x4 and 1x5. In addition, the authors alter the method of channel-attention to channel self-attention. Alternatively to using global average pooling the authors extract 32 features from each channel using average pooling and then learn attention scores within each channel, instead between each channel. Interrelation among channels is captured through sharing the weights of the attention operation across all channels. Furthermore, the authors extend the model output with a technique, called attention-pooling [24], and alter it to Multi-head Attention, which pools the final feature vectors into a feature space of size $N \cdot L$, where N is the number of predicted classes and L are feature aggregations. The inputted feature vectors are subdivided into N segments. The attention block uses two trainable weights matrices U and W that linearly project the feature vectors and apply softmax to scale the weights. The scaling values are multiplied with the inputted feature vector. However, the paper [37] does not state in detail the shape and size of the inputted feature maps from the previous channel-attention layer. Based on the above approaches about residual Convolutional Networks, Multi-head Attention, Squeeze and Excitation Network and the Inception Network it is not possible to reason, which of these approaches are the most relevant mechanism that lead to an improved performance within

the challenge. Therefore, this thesis analysis each of these approaches, especially the Multi-head Attention mechanism, by investigating each approach independently and propose modifications to them for the multi-label classification of 26 cardiac arrhythmias, which will be described in the sections 3 and 3.3.

2.2 Related work for the classification of ECGs based on Transformer models

In recent years, Transformer models have gained considerable popularity because their model and training design can be much more easily accelerated on a graphic processing unit (GPU), due to the capability of parallel weight updates during backpropagation through shared attention (query, key and value) matrices. This leads to a significant reduction in training time and hardware costs, while maintaining comparable or even better performance, compared to the previous Long-Short-Term-Memory (LSTM) [17] or Recurrent Neural Network (RNN) [20], which were widely used until Transformer models for sequential data. LSTMs and RNNs need to process data sequentially through their units and do not scale well for large training applications leading to computational slowness and costs. In addition, Transformer models have the ability to process long-range sequential data much better, because LSTMs and RNNs can suffer from long-range dependencies caused by vanishing gradient problems during backpropagation. Choi et al. [6] propose a more complex encoder-based Transformer model and training methodology for several ECG processing tasks, called ECGBERT, that closely models the architecture and training procedure of BERT [8]. BERT, based on the Transformer encoder architecture, was one of the first successful and popular language models fine-tuned on various tasks. Following the architecture design of BERT, ECGBERT is fine-tuned on multiple down-stream tasks, i.e. Atrial Fibrillation classification (binary), heart-beat classification (normal, unknown, supraventricular ectopic, ventricular ectopic and fusion heartbeats), user verification by using the ECGs as biometric authentication to predict patient IDs and lastly sleep apnea detection. Similarly to BERT, the authors create an own vocabular, proposed as wave segment vocabular, to tokenize each ECG and apply then BERT's training procedure "Masked Language Modeling" (MLM) to pre-train their model. In a preparation step, the ECG signals are pre-processed (normalised, filtered etc.) and splitted by fiducial points into wave segments. Time-frequency analysis and Hamilton's algorithm [29] are applied to clean the ECG signals and extract and split by onset and offset points. The obtained wave segments are used to cluster the waves using K-mean and Dynamic Time Warping. For this, the authors train four classifiers, each classifier correspond to one the waves P, QRS, T or background waves. 70 clusters are obtained with 12 P, 19 QRS, 14 T and 25 background (e.g. PR or ST intervals) wave clusters each. Next, each segment is classified by the corresponding classifier and assigned to a wave cluster. The classified wave segments are then mapped to predefined tokens. The authors do not describe how the tokens are encoded, an assumption could be that these are random initialized embeddings. In addition, CNN features are extracted from the raw ECG using an U-Net architecture [34]. The authors reason that the model can only learn high-level wave context relations from the ECG tokens, but CNN features provide refined pattern information and therefore are crucial for extracting fine-granular features. The tokens are combined with CNN feature embeddings obtained from the U-Net and positional information is added, likewise in the original Transformer paper [1]. Based on this, the authors create training samples from the ECGs that form sentences of wave segments, where each sentence contains 1-8 consecutive heartbeats and a heartbeat is composed of several wave segments. The constructed ECG wave sentences are then inputted to the Transformer-encoder module, where several sequences can be inputted, which are splitted by a separation "[SEP]" token. The

model is pre-trained using 15% masking, similarly to Masked Language Modeling in BERT. For the pre-training phase of the model the authors use the MIT-BIH database (AFIB/arrhythmia) [13] and the Apnea-ECG database from Physionet. Based on the pre-trained model weights the authors use the model to fine-tune it on several down-stream tasks by adapting the output layer to the corresponding tasks. The model achieves for binary classification of Atrial Fibrillation an accuracy score of 97% and for beat classification about 86% on an evaluation set based on the MIT-BIH database. Another work from Zhao [50] lists in a review from 2023 several Transformer-based models for ECG classification. Many of the Transformer-based ECG models discussed in the review combine the Transformer encoder module with a Convolutional Network [3]. [5] [9] [22] [26] [50]. The author argues that a Convolutional Network has a limited receptive field, which prevents the model from learning distant dependencies and extract local patterns. On the other hand, Transformers can learn long-range dependencies through their attention mechanism. The combination of both can capture spatio-temporal information from the ECG signal [50]. Hu et al. [22] propose an encoder- and decoder-based Transformer architecture. First, the model extracts features from a single-lead ECG signal using multiple convolutional layers. The authors add positional encoding to each feature map and use them as inputs to a Transformer encoder and decoder block, similar to the original Transformer architecture [1]. The decoder block unmasks each feature map by feature map, through which the decoder block sequentially classifies the next ECG segment. The paper uses the MIT-BIH arrhythmia database [13], which contains annotations for each heartbeat (Normal, Ventricular, Supraventricular, etc.). Bing et al. [3] propose an encoder-based Transformer architecture that combines a Vision Transformer with a Convolutional Neural Network, called ConVit. Vision Transformer (ViT) [10] was one of the first papers that effectively applied Transformer models to computer vision tasks. In Vision Transformers an embedding is obtained by a subpart/pixel group from an image and the model extract global relations within the image for classification task by considering relations within image subspaces. A speciality about Vision Transformers is that it do not apply convolutions, although the model shows state-of-the-art performances within computer vision tasks.

2.3 Related work for the specific classification of SR, AF, AFL, PAC and PVC based on deep-learning models

Wang et al. [43] analyse in their paper the particular detection of Premature Ventricular Contractions and propose a Convolutional Network, called PVCNet. The authors apply a deep Convolutional Network in combination with several feed-forward layers. The input of the model is a short-term single-lead ECG. The authors use the publicly available MIT-BIH database [13] for pre-training and test it on the St. Petersburg INCART database. Their model is able to achieve an accuracy of 98% on the test set. In their work, the authors highlight the importance of several pre-processing steps, including data augmentation to overcome class imbalance, splitting the training and validation sets by patients, ECG resampling and filtering, i.e. a butterworth band-pass filter, as the authors justify that this filter can retain the main frequency components of the signal. Li et al. [18] highlight in their paper the importance of improving the accuracy of assistive ECG devices. In their work they develop a deep-learning approach that specifically classifies Atrial Fibrillation, Premature Atrial Contractions and Premature Ventricular Contractions from the first two leads of an ECG. In the study the authors find that their assistive ECG device makes about 18%-24% false positives after revisiting the labels. In their approach the authors use a combination of a Convolutional Network and a Long-Short-Term-Memory (LSTM). In the first stage, the CNN model extracts features with the P, QRS and T intervals. The extracted features are then fed into a LSTM model, which successively classifies the extracted features. The authors

justify that the interaction between the two models ensures that spatial-temporal features are extracted. With this approach the authors are able to increase the accuracy compared to the device from 0.77 to 0.86 (AF), 0.76 to 0.84 (PVC) and 0.82 to 0.87 (PAC), after manually revisiting labels. Another study by De Marco et al. [7] focuses on the classification of non PVC and PVC using only the QRS complex from an ECG. For this the authors utilize the MIT-BIH database [13] by extracting all QRS complexes from long-term ECGs in the MIT-BIH database. In their study they compare several models, e.g. random forest, LSTM, bidirectional LSTM, ResNet-18, MobileNetv2 and ShuffleNet. MobileNetv2 [36] and ShuffleNet [49] are Convolutional Networks that have been designed for efficiency and to work on mobile devices. Ribero et al. highlight in their work [32] and [33] that the classification of Atrial Fibrillation and Atrial Flutter is not a trivial task. In their paper, the authors propose a 1D and 2D Convolutional Network trained on 1D ECG signals and 2D ECG images. The underlying datasets are a combination of the Physionet 2021 challenge database and private datasets collected from various cardiologists and hospitals. In their study [33] the authors state that the performance of the model trained on only the Physionet 2021 challenge data is much lower than that trained on their own dataset, which might be due to poor data quality of the Physionet 2021 challenge data as the authors justify. Wang [42] investigates the classification of Atrial Fibrillation and Atrial Flutter and proposes a combination of a Convolutional Network with an improved version of the Elman Neural Network (IENN) [12], a specific form of the Recurrent Neural Network (RNN) architecture, which uses a context node for improved contextual memory. The combined model is able to achieve around 99% accuracy on the MIT-BIH [13] database for the binary classification problem. However, a drawback of the study [42] is that the model requires large and various ECG data to be trained. Another limitation of the studies mentioned in this section is that the models are limited to specific arrhythmia types.

Chapter 3

Methods

This chapter discusses the utilized and modified approaches developed for this thesis. It is divided into three sections. The first section discusses the mathematical background of Transformer models. It highlights the architectural differences between a Transformer model trained on language and applied to ECG signals. The second section gives a brief overview of the main components of a Convolutional Network. The third section presents the modified approaches that are used to address the problem of multi-label ECG classification into 26 different arrhythmias, here a feature-based random forest classifier, a residual Convolutional Network with and without dilation, a Transformer encoder-based model, a multi-scale Convolutional Network, a Squeeze and Excitation Network and the proposed approach, an ensemble of a multi-scale Convolutional Network with several Transformer encoder blocks. Finally, section 3.3.8 describes how the models are fine-tuned on the Physionet 2021 challenge data. All model implementations developed in this thesis can also be found in this Github repository.

3.1 Background about Transformer models

The Transformer model was introduced in the paper "Attention Is All You Need" by Vaswani et al. [1]. Figure 3.1 shows the model architecture.

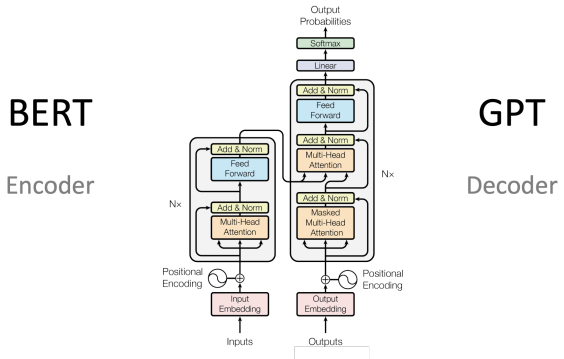

Figure 3.1: Transformer architecture
Source: "Attention Is All You Need" [1]

The model consists of two parts, the encoder block, left half in 3.1, and the decoder block, right

half in 3.1, connected by the arrow in the middle. Initially, this architecture was designed for language translation. The encoder block receives the input sequence and encodes the words (tokens). The encoder block maps then the input sequence to the translated output sequence via the decoder block, which predicts the next most likely output token. The main strength of the Transformer model is the "attention mechanism" implemented in the "Multi-Head Attention" block 3.1. The Multi-head Attention block projects and merges each feature embedding with all others features embeddings in the input sequence. The objective of the attention block is to learn relational attention weights among the embeddings and merge and scale the input embedding sequence based on these weights. In comparison, Convolutional Networks aim to extract invariant features from local inputs and feature groups. Depending on the kernel size a convolutional filter applies only to local surrounding input and feature groups, while the attention block applies to the entire input sequence as whole. For this, the attention block computes query, key and value pairs by considering each input element with each other input element individually. The query, key and value pairs are used to learn attention weights. The attention block adapts during training the attention weights to different arrangements in the training data, i.e. it extracts relationships and learns to attend to the most common relations and features from various data arrangements. In section 3.1.2 the attention mechanism is in more detail discussed. Many researchers have adopted the original Transformer architecture and use parts of it for various downstream tasks, such as (text) classification tasks, e.g. BERT (Bidirectional Encoder Representation from Transformers) [8], or for regression tasks (text completion), such as GPT (Generative Pre-trained Transformer) [4]. BERT uses only the encoder block, while GPT uses only the decoder block. This thesis utilises and implements only the encoder block of the Transformer model. Although, the computation in both blocks are quite similar, the decoder block is designed for sequence completion tasks and uses for the attention operations the output of the encoder block, also referred to as cross-attention. Based on the input sequence and the output of the Transformer encoder block, the decoder block sequentially unmasks and predicts the next token from the output sequence. Through the masking operation the attention in the decoder block focuses only on unidirectional states. Whereas the encoder block needs one input sequence and computes self-attention on the entire sequence bidirectional. This design makes the decoder less suitable for classification tasks and could prevent a classification model from learning a more meaningful representation from the entire ECG, because valuable information would be masked. Therefore, this thesis investigates an encoder-based Transformer model. In general, a Transformer model divides the input sequence into subspaces, called embeddings. Embeddings are vectors that represent specific input features. In language-based Transformers, each embedding represents an assigned subword or character in the known vocabulary of the model, also called tokens. The input preparation process for language-based Transformers work different as in this thesis and is usually handled by a tokenizer, a separate sequence mapping and vocabulary indexing tool of the model. Tokenizers are one of the key differences between Transformers trained on language tasks and Transformers trained on ECG signals. In this thesis, the Transformer model does not utilize a tokenizer to map the ECG signal. One reason is that the signals are already present as processable numbers and do not need to be encoded. The other reason is that words in language are repetitive, whereas heartbeats or segments within ECG signals have unique amplitudes. By indexing these into a finite feature space of arrhythmia features could possible prevent the model from learning detailed features. Therefore, this thesis experiments with both raw ECGs and features obtained from convolutional filters as input to the Transformer encoder block. This includes either splitting the signal into fixed-sized segments or extracting features maps / channels with a Convolutional Network. This will be analysed and discussed in the evaluation 4.2 and discussion 5. The input size of a Transformer model needs to be fixed-sized, also known as the context window of the model. In this thesis, this is

either a 10 or 60 seconds long single-lead ECG signal, depending on the investigated experiments (either Physionet 2021 or MyDiagnostick data). The ECG embedding features are fed directly into the encoder block, whereas in language-based Transformer models an additional trainable embedding matrix represents the first layer of the model before the encoder block.

3.1.1 Positional encoding

The original Transformer model adds positional information to each embedding before these are passed into the attention block. The positional encoding layer is connected with the output of the attention block via a residual connection. This ensures that temporal information about the order of the embedding sequence is preserved and propagated throughout the network. The original paper proposes a positional encoding technique using sine and cosine functions of different frequencies:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right) \tag{3.1}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right) \tag{3.2}$$

"Pos" represents the index position of each embedding in the sequence and i is the embedding dimension, which is equal sized for all input embeddings. Each dimension in the embedding corresponds to a single sinusoid and the wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$ that projects the embedding values into a positional order. The authors chose this function, because they hypothesised that it would allow the model more easily to learn and attend to relative positions, since for any position k, $\mathrm{PE}_{\mathrm{pos+k}}$ can be represented as a linear function of $\mathrm{PE}_{\mathrm{pos}}$. [1]. This thesis investigates the effect of adding positional information to ECG segments for arrhythmia detection in ECGs and discusses this in the evaluation 4 and discussion 5 sections.

3.1.2 Attention mechanism

The input to the Transformer encoder block is a sequence of length N, consisting of N positional encoded embeddings (features) with dimension d, i.e. a matrix of the shape $N_{sequence\,length} \cdot d_{embedding\,dimension}$. The block uses three trainable query, key and value matrices, denoted by the matrices $\mathrm{W^Q}$, $\mathrm{W^K}$ and $\mathrm{W^V}$, for projecting the input sequence into three new feature spaces. The multiplication of the input sequence with each weight matrices $\mathrm{W^Q}$, $\mathrm{W^K}$ and $\mathrm{W^V}$, yields the matrices Q, K and V that go into the attention block, which are represented by the three arrows in 3.1. An important advantage is that the weights $\mathrm{W^Q}$, $\mathrm{W^K}$ and $\mathrm{W^V}$ can be computed in parallel during the forward and backward pass of the model and do not depend on sequential states as in LSTMs or RNNs. The next steps form the core of the Transformer model, which is the "scaled dot-product attention" inside the Multi-head Attention block 3.1, shown in figure 3.2 and formula 3.3. This sections explains the calculation of the attention mechanism, while the next section 3.1.3 explains the extension to Multi-head Attention. The first step in the Multi-head attention block is to compute the dot-product between Q and K by multiplying Q with K^{T}, which gives an unscaled attention weight matrix A. Although, A is not shown in 3.2, it is used for explanation here and refers to the first output of the bottom purple "MatMul" block, that is Q multiplied with K^{T}. Since every projected embedding in Q is multiplied with every transposed projected embedding in K^{T}, the size of the attention weight matrix A is $N_{sequence\,length} \cdot N_{sequence\,length}$, which is the quadratic size of the inputted sequence. Therefore, a disadvantage of Transformer models is that the trainable projection weights $\mathrm{W^Q}$, $\mathrm{W^K}$ and $\mathrm{W^V}$ scale quadratic with the length of the input sequence. The obtained attention

matrix A represents how much each embedding relate to each embedding in the same sequence (self-attention), which is learned through adapting W^{Q}, W^{K} and W^{V}. The next step, which is shown through the yellow "Scale" block in 3.2, normalizes the attention values of A by a normalising constant $\sqrt{d_k}$, which also corresponds to the denominator in formula 3.3. $\sqrt{d_k}$ is the size of the column dimension of W^{K}, where W^{K} is of size $d_{embedding\,dimension} \cdot d_k$, more precisely $N_{heads} \cdot d_{subembedding\,dimension} \cdot d_k$, but which is discussed in more detail in the next section about Multi-head Attention 3.1.3. This operation helps the model to reduce the variance of the computed relational values from Q and K and stabilizes training, as proposed by the authors [1]. The masking operation (optional) 3.2 is skipped here, because it is only used in the decoder block 3.1. In short, the masking operation adds triangular matrix masking to the attention matrix A so that the further processed attention values depend only on the unmasked embeddings. Next, the attention weight matrix A is scaled by a softmax function to obtain softmax-normalised and interpretable weights, shown through the green box in 3.1.3.

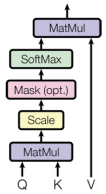

Figure 3.2: Scaled Dot-Product Attention
Source: "Attention Is All You Need" [1]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.3}$$

Finally, the softmax-normalised weights of A will be used to scale the projected value matrix V, shown in 3.2 as the top purple "MatMul" box. This operation aims to project the learned relational embedding attention weights from A into V by matrix multiplication. Previously, V represented a projected feature space of the input embedding sequence, while the final multiplication with A scales V into a desired projected feature space based on the attention weights. The obtained embedding matrix V represents a contextualized version of the inputted embedding sequence. Afterwards, which is not shown in 3.2 but in 3.3, another linear matrix multiplication W^{O} follows to the scaling operation of V as final output. This matrix operation, which should not be interchanged with the additional feed-forward layer as part of the Transformer encoder block 3.1, is a trainable linear weight matrix that projects the embedding sequence again, obtained from the concatenated heads. Since the outputted embedding sequence matrix from the Transformer encoder block should have the same size as the inputted embedding sequence, i.e. $N_{sequence\,length} \cdot d_{embedding\,dimension}$, the initial shape can be restored by this linear matrix operation. The dimensions of W^{O} depend on the size configurations of W^{Q}, W^{K} and W^{V}. This is also discussed in more detail in the next section 3.1.3. It allows that the outputted embeddings from the Multi-head Attention block can be added with the inputted embeddings through a residual connection shown in 3.1, which requires the same input and output sizes for the summation operation. The residual connection ensures that the initial features and temporal information is propagated throughout the entire network. In addition, layer normalisation is applied to stabilise the weights during training 3.1. Next, a feed-forward layer follows the attention block 3.1. The feed-forward layer allows the model to learn non-linearity, since the

operations inside the Multi-head Attention block remain linear matrix multiplications. In the original paper one hidden-layer with relu activations is used [1]. The output of the feed-forward layer is then reconnected by a second residual connection and again normalised 3.1. Since the output of the Transformer encoder block has the same size as the input, i.e. $N_{sequence\,length} \cdot d_{embedding\,dimension}$, it allows to stack multiple Transformer encoder blocks, where each block can learn and attend to different relations among the inputted embedding sequence.

3.1.3 Multi-head Attention

In addition, the original Transformer model introduces a mechanism called "Multi-head Attention" [1]. The idea of Multi-head Attention is to train multiple attention matrices on different inputted embedding subspaces simultaneously in the same attention block. This allows to capture a wider range of relationships and to encapsulate partial information from the embeddings into partial contexts. Figure 3.3 illustrates the entire Multi-head Attention process.

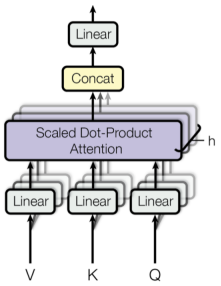

Figure 3.3: Multi-head Attention
Source: "Attention Is All You Need" [1]

Multi-head Attention splits each embedding into h sub-embeddings before these are passed in parallel to h W^Q, W^K and W^V matrices. In the original paper 8 heads were used [1]. Following the example from the original paper, an attention layer, which receives $N_{sequence\,length} \cdot 512_{embedding\,dimension}$ as input, splits each embedding into 8 sub-embeddings, which yields $N_{sequence\,length} \cdot 8_{heads} \cdot 64_{subembedding\,dimension}$, as input for the attention layer. This can be thought by slicing horizontally the input sequence with vertical feature embedding vectors in 8 pieces to obtain 8 parallel partial embeddings sequences as input for the Transformer encoder, where each sequence is separately passed to another (smaller) attention block that work in parallel, also referred to as "heads". The dimension of the sliced sub-embeddings is here described as $qkv_{dimension}$, where $N_{heads} \cdot qkv_{dimension}$ should normally yield the initial inputted embedding dimension $d_{embedding\,dimension}$. Moreover, $d_{embedding\,dimension}$ should be divisible by N_{heads} to obtain equal-sized attention blocks. The computation then proceeds as described in the previous section about the scaled dot-product attention, but here in parallel on different embedding subspaces and for multiple smaller sized W^Q, W^K and W^V matrices. After the computation within the attention block the outputted sub-embeddings are then concatenated again and multiplied by a trainable linear layer, shown in figure 3.3 as "Linear" at the top. The multiple "Linear" blocks at the bottom simply represent the reshaping procedure of Q, K and V into h heads. However, the "Linear" operation at the top represent trainable weights that serve as a linear projection operation, without the use of an activation function, which weighted-based concatenate the sub-embeddings from the heads. In addition, it can ensure that the initial input shape of $N_{sequence\,length} \cdot d_{embedding\,dimension}$ is restored, even if the shape after "Concat" does

not match the initial shape. Generally, the input and the output dimension of the standard attention block has an equivalent shape of $N_{sequence\,length} \cdot d_{embedding\,dimension}$. This means that $N_{heads} \cdot qkv_{dimension}$ is not necessarily equivalent to the embedding shape $d_{embedding\,dimension}$, thus the attention block can be of variational size and learn attention weights on a densed embedding feature space. In addition, it serves as a trainable linear projection operation that concatenate the sub-embeddings received from the heads.

3.2 Background about Convolutional Networks

Convolutional Networks are often applied deep-learning models for the classification of ECG signals [18] [7] [43], including the first four ranked models and many other models within the Physionet 2021 challenge [27] [15] [44]. Traditionally used for computer vision tasks, a Convolutional Network can similarly be applied to one dimensional data. The main component in a Convolutional Network is the convolutional filter (also called kernel), which is a fixed-sized window that stripes over the input data, where each data point and its surrounding data points within the window are multiplied with the kernel weights and added to project the data points into one new outputted data point. The kernel then stripes over the entire input data. The outputted projection is called feature map or channel. At the beginning the kernel weights are randomly initialized, where the kernel later adapts and optimizes the weights to features in the training data by learning patterns, such as corners or edges to more abstract concepts like faces in computer vision tasks. Convolutional filters are also called spatially invariant feature extractors, meaning that it does not matter where the features are located in the input data, they will still be extracted by the convolutional filter. Commonly, several convolutional filters work in parallel within the same convolutional layer, yielding several outputted features maps or channels that are stacked in depth. Therefore, a convolutional layer is typically characterized by the number of convolutional filters, kernel, stride and padding size. The number of filters in a layer determines how many distinctive patterns the convolutional layer can learn within a specific network level from the data, while each feature map contain different features from the layer. Usually, the number of filters starts small, e.g. 32 and the number of filters increase over consecutive layers, e.g. 64, 128 and so on. The reason for this is that in deeper layers more abstract concepts need to be learned by combining simpler corner and edge features from previous layers. This process requires a larger number of filters to capture the increasing complexity and diversity of features. The filter size describes the width and/or height of the filter window and mainly influences the number of trainable parameters. Smaller filters can focus on details, while larger filters can receipt distant associated patterns. Smaller windows may also prevent overfitting and enable deeper and more complex network hierarchies. Stride determines the number of data points the kernel window is shifted when sliding over the input data. Therefore, some data points can be left out by skipping. Padding adds margin to the borders of the input and can reduce the input size. Stride that is larger than the kernel size and padding can miss relevant input information by ignoring or skipping features, but reduce the feature space and number of trainable parameters. Beside standard convolutional filters, stride and padding, dilated convolutions can be applied. Dilated convolutions employ a kernel that incorporates strides. These kernels extend the receptive field and extract features with less resolution. Figure 3.4 illustrates a dilated convolution applied on one dimensional ECG data. This thesis investigates the effect of dilated convolutions for the classification of ECGs. Typically, a convolutional layer is followed by a pooling layer that aims to reduce dimensionality. Other than the convolutional layer, the pooling layer uses a single kernel across all channels that (most commonly used) applies "min", "max" or "average" pooling. This means, the pooling kernel forwards either the minimum, the maximum value or

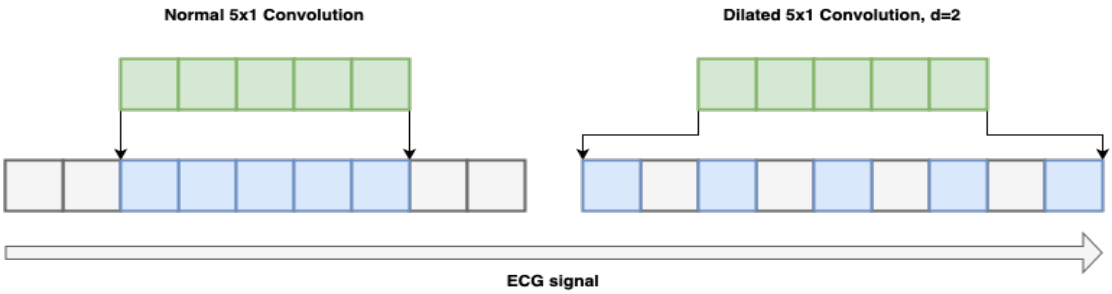

Figure 3.4: Dilated convolution

averages the values within the window. For example, in a 1D signal a pooling window size of two will reduce the input dimensionality by half, thus for two data points a single data point is outputted and forwarded to the next layer. This process enables the convolutional network to extend the receptive field and learn from detailed features to more widely scattered features. As an alternative to pooling layers strides can be used for downsampling the input. Other common components in Convolutional Networks include feed-forward layers using non-linear activations. These layers are utilized to learn non-linear features, since convolutional kernels only apply linear multiplication operations. The final layer of a Convolutional Network commonly apply global pooling, that denses all values within a channel into a single value, e.g. min, max or average, from which then a feed-forward or softmax layer follows as the final model output, depending on the task.

3.3 Approaches

3.3.1 Feature-based Random Forest

As a baseline comparison a feature-based random forest classifier is trained. This thesis uses the Sklearn Python framework, which consists of an in-built implementation of the random forest and the Biobss Python package to extract features from ECGs as input for the random forest classifier. A random forest is an ensemble of a predefined number of decision trees that use majority voting or averages to enhance the overall performance. The random forest in this thesis uses 100 decision tree estimators. The model can be used for both, regression or classification tasks. The individual decision trees, as done in this thesis, can be trained by bootstrap samples of the training data. A decision tree is a hierarchical tree structure, which consists of one root node, internal nodes, branches and leaf nodes. Here the inputted features are continuous numerical variables, which will be described in a bit. The nodes are hierarchically connected by branches. During training each tree iteratively learns a set of decision boundary nodes (thresholds) based on the bootstrapped training data to minimize the overall classification error. Various methods are available, e.g. Entropy, Information Gain and Gini Impurity. Here Gini Impurity is used, which is a common applied method. Each node represents a decision boundary for a specific feature that best partitions the samples into classes. This depends on the lowest calculated Gini Impurity value for each feature. The formula for Gini Impurity is given below 3.3.1:

$$G(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{3.4}$$

The method creates the decision tree structure by calculating for each feature in the bootstrapped training sample the Gini Impurity, while the feature with the lowest Gini Impurity value is taken as next decision node, starting from the root node. From there the Gini Impurity is recursively calculated for the remaining features for the next decision nodes, until for each tree path a leaf node is reached. A termination of a path (leaf node) is reached, when all samples that end up in this node belong to the same class, such that no further partitions by a decision node is needed. Another termination condition is reached, when no more remaining features are available in a specific tree path, since the same feature partition can not occur more than ones in the same path. The Sklearn Python framework provides here several hyperparameters to configure this, e.g. maximum depth of each tree. The Gini Impurity 3.3.1 is calculated for each (remaining) feature in each path and step, when adding the next decision node. For categorical features it takes the data samples D that end up in this node (in the root node this corresponds to all bootstrapped samples) and calculates for each class c the probability p that a feature value corresponds to a specific output class. Each probability is squared and the sum of all squared probabilities is subtracted from 1. The feature that most partitions the samples into a specific class in a node shows the lowest Gini Impurity value. For numerical features this involves some more steps, since multiple Gini Impurity values within each feature are calculated. For numerical variables all sample values for one feature are sorted and averages from consecutive sorted value pairs are used as thresholds to calculate the Gini Impurity. The lowest calculated Gini Impurity value of all thresholds within a feature is then used as Gini Impurity value for the entire feature. The threshold is used as decision boundary for the next node, when this feature has compared to all other features the lowest Gini Impurity value. This process happens recursively until in each tree path all leaf nodes are reached. In this thesis no feature selection is carried out for the random forest, because the model is less sensitive to irrelevant features. This is due that non-informative features are downstreamed behind relevant features and end up in the bottom nodes. However, if some bootstrap samples are unlucky chosen, some decision trees might put irrelevant features in the top nodes, so feature selection could be considered.

The Biobss library provides a set of toolkits for extracting features from ECG signals. Table 3.1 shows 39 extracted features using the Biobss library and a short description for each feature. In the first step, the Biobss library calculates all R peaks and other fiducial points in the ECG, e.g. P, Q, S and T wave onset and offset points. For the point localisation the package has one of these three methods implemented: "pantompkins" [29], "hamilton" [14] or "elgendi" [11]. For the experiments the "pantompkins" method is used, because according to the research work by Khan and Imtiaz [25] that compare different fiducial points detection methods, it is the most widely used approach for RR peak detection with high accuracy. In the next step, inbuilt functions in the Biobss library calculate 39 morphological features from the extracted ECG locations (peaks and wave onset/offset points). The Biobss library splits the ECG into segments using the detected fiducial points, where for each segment peak amplitudes, intervals and ratios are calculated as features. The library provides two functions: biobss.ecgtools.ecg_features.from_Rpeaks and biobss.ecgtools.ecg_features.from_waves. "from_Rpeaks" takes as reference four consecutive R peaks and calculates the corresponding (current) R peak amplitude, RR intervals (the RR interval before the current R peak and two RR intervals after the current R peak), ratios and the mean of the intervals (3.1, row 1-8). "from_waves" takes as reference two consecutive R peaks and calculates the corresponding P, Q, R, S and T amplitudes, their intervals and ratios (3.1, row 9-39). Each feature is an average from all segments in the ECG signal. After feature extraction some feature vectors had "nan", "-inf" or "inf" entries, which have been manually imputed by means. The features are then used to train a random forest classifier. The classifier results are discussed in section 4 and 5.

Biobss features names	
Features (averaged)	**Description**
a_R	Amplitude of R peak
RR0	Previous RR interval
RR1	Current RR interval
RR2	Subsequent RR interval
RRm	Mean of RR0, RR1, and RR2
RR_0_1	Ratio of RR0 to RR1
RR_2_1	Ratio of RR2 to RR1
RR_m_1	Ratio of RRm to RR1
t_PR	Time between P and R peak locations
t_QR	Time between Q and R peak locations
t_SR	Time between S and R peak locations
t_TR	Time between T and R peak locations
t_PQ	Time between P and Q peak locations
t_PS	Time between P and S peak locations
t_PT	Time between P and T peak locations
t_QS	Time between Q and S peak locations
t_QT	Time between Q and T peak locations
t_ST	Time between S and T peak locations
t_PT_QS	Ratio of t_PT to t_QS
t_QT_QS	Ratio of t_QT to t_QS
a_PQ	Difference of P wave and Q wave amplitudes
a_QR	Difference of Q wave and R wave amplitudes
a_RS	Difference of R wave and S wave amplitudes
a_ST	Difference of S wave and T wave amplitudes
a_PS	Difference of P wave and S wave amplitudes
a_PT	Difference of P wave and T wave amplitudes
a_QS	Difference of Q wave and S wave amplitudes
a_QT	Difference of Q wave and T wave amplitudes
a_ST_QS	Ratio of a_ST to a_QS
a_RS_QR	Ratio of a_RS to a_QR
a_PQ_QS	Ratio of a_PQ to a_QS
a_PQ_QT	Ratio of a_PQ to a_QT
a_PQ_PS	Ratio of a_PQ to a_PS
a_PQ_QR	Ratio of a_PQ to a_QR
a_PQ_RS	Ratio of a_PQ to a_RS
a_RS_QS	Ratio of a_RS to a_QS
a_RS_QT	Ratio of a_RS to a_QT
a_ST_PQ	Ratio of a_ST to a_PQ
a_ST_QT	Ratio of a_ST to a_QT

Table 3.1: 39 Biobss features

3.3.2 Encoder-based Transformer

All developed deep-learning models, starting from this section, have been implemented with the Python framework TensorFlow 2. As second investigated model a standard Transformer encoder model is utilized. Two types of the Transformer model are investigated within this thesis. First a "standalone" Transformer encoder-based model and secondly a modified version of the Transformer encoder model is proposed that uses a Convolutional Network based on several residual blocks of multi-scale convolutions as input features for the Transformer encoder and which will be described in section 3.3.6. The main parts of the Transformer encoder block applied in this thesis are similarly to the standard Transformer encoder block from the original paper [1] and as described in section 3.1 and shown figure 3.1. The Transformer encoder block is composed of a positional encoding layer, a Multi-head Attention layer, a layer-normalization layer, a feed-forward and layer-normalization layer with residual connections in between. Since the Transformer encoder model is intended to learn relational features from the embedding sequence, the version of the Transformer encoder model described in this section segments each ECG into fixed-sized embeddings. ECGs with different recording lengths are zero-padded or truncated respectively, which will be discussed in 4.1.3. For example, an ECG with 2000 datapoints is reshaped into 40 segments each with 50 datapoints. This operation refers to the inputted embedding layer in 3.1. The positional encoding layer that adds temporal information to the embeddings is an optional layer, which is conducted in the experiments 4.2. In the experiments different parameter configurations of the Multi-head Attention block are analysed, e.g. the performance of stacking several encoder blocks.

3.3.3 Residual Convolutional Network with and without dilation

As third model two residual Convolutional Networks are investigated based on the idea of ResNet [16]. Both networks consist of 6 residual blocks. The residual blocks are composed of two 1D convolutional layers with kernel size 5x1, strides 1 and no padding, where the channels from the first convolution layer are connected via a skip connection to the last layer of each block. The second convolutional layer is followed by a batch normalization layer, a feed-forward layer using leaky relu activations for extracting non-linear features and a dropout layer with rate 0.1. The final output of the dropout layer is connected by the skip connection with the first convolutional layer. The number of channels increase per residual block with 16, 32, 64, 128, 256 and 512. Furthermore, between each residual block a 1D average pooling layer with stride 2 is applied, which denses the features after each residual block. Finally, the model is concluded by a global average pooling layer and a dense layer to the number of outputted classes using sigmoid activations. Sigmoid activations are used in preference to a softmax output layer, which is commonly used for classification tasks, since the data is multi-label annotated and each class need an outputted probability score. The corresponding model graph is given in figure graph 3.5. The second residual Convolutional Network experiments with dilated convolutions using similarly a kernel size of 5x1, although with a dilation rate of 2.

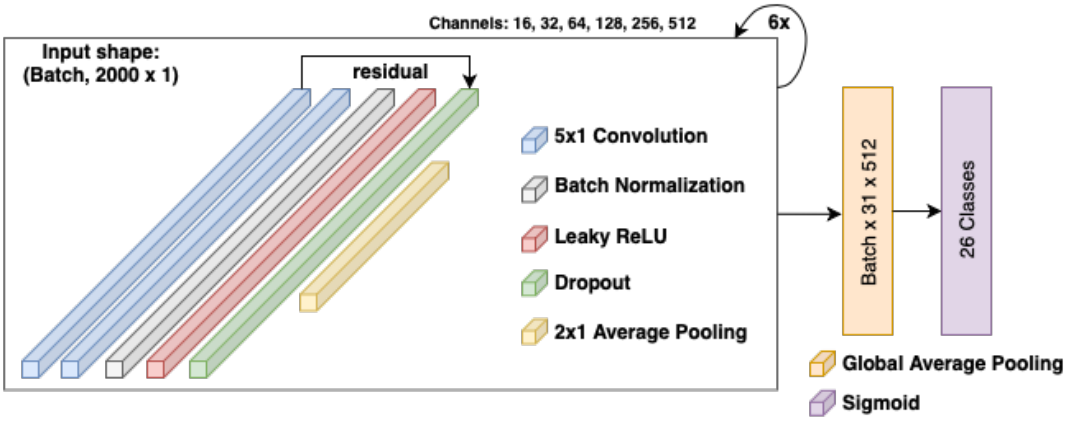

Figure 3.5: Residual Convolutional Network

3.3.4 Multi-scale residual Convolutional Network

The fourth investigated method adopts the idea from the Inception Network. "Inception" proposed by Szegedy et al. in "Going deeper with convolutions" [38] set new state-of-the-art performance on the ImageNet classification challenge [35] in 2014, while it is a 20 layers deep network it uses much fewer trainable parameters. The idea of inception is to use several convolutional filters in parallel in the same convolutional layer, e.g. in the original paper 1x1, 3x3 and 5x5 have been applied. The outputted channels are then concatenated into one feature map block for further processing. The idea behind 1x1 convolutions is that it reduces dimensionality. Unlike pooling, which takes for example the minimum, maximum or average values within a channel, 1x1 convolutions apply dimension reduction across channels by learning a set of weights equal to the number of channels. The expression 1x1 convolution might be a bit misleading, because more precisely it uses a $N_{number\,of\,channels}$x1 convolution that applies to channel depth in which each channel value is weighted-based summed from all channels into one channel. This requires that all feature maps have the same sizes. The overall strength in the inception mechanism is extracting and combining features from different kernel sizes, while keeping the dimensional overhead low using 1x1 kernels. Although, the term "attention" is not directly used in the paper [38], the 1x1 convolution might be compared to an attention operation, since a set of linear weights is learned that weight channel information, while channel-attention, which is described in the next section 3.3.5, incorporates non-linear activation functions, e.g. relu and sigmoid, to weight channels based on a set of learnable weights. This thesis makes a few modifications to the proposed Inception Network [38]. The modified version in this thesis is motivated by the idea of combining small and large sized kernels, i.e. 3x1, 5x1, 9x1 and 15x1 in parallel, thus a wider range of features can be captured from these multi-scaled kernels. The modified version consists of 4 blocks and is shown in figure 3.6. The channels are equal sized in dimension and added into one stack in the concatenation step.

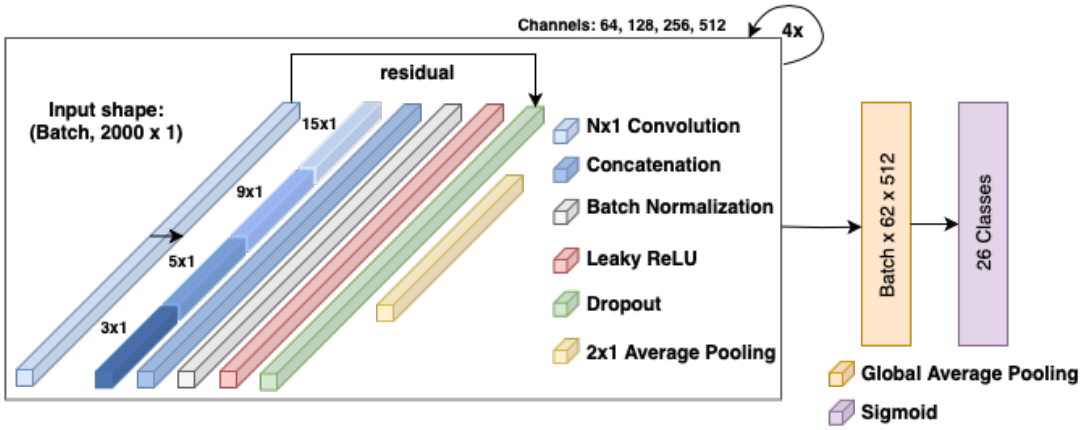

Figure 3.6: Multi-scale residual Convolutional Network

3.3.5 Squeeze and Excitation Network

The fifth investigated method is channel-attention implemented through a network architecture called Squeeze and Excitation Network (SENet), proposed by Hu et al. in [21]. The model set new state-of-the-art performance on the ImageNet classification challenge [35] in 2017. The application of this model is motivated, as it uses a related variant of an attention mechanism and is incorporated within the residual Convolutional Network of the second and fourth ranked model from the Physionet 2021 challenge [15] [37]. The idea of channel-attention is to recalibrate the channels of a convolutional layer. This is achieved by learning in a parallel "Squeeze and Excitation Network" a set of attention weights. In a first step, the Squeeze and Excitation Network "squeezes" all channels from the previous convolutional layer by applying global average pooling. The channel values are then fed into a two hidden layer feed-forward network using relu activations in the first layer and sigmoid activations in the second layer. The network is intended to learn inter-dependency between the channels. In the original paper [15] the authors experiment with the size of the hidden layer. They use a dense ratio to the number of channels to evaluate the performance of the network and which has been empirically determined, e.g. 2-32, while smaller ratios increase the block sizes. The authors find that a ratio of 16 offers a good balance between accuracy and model complexity. The dimensionality is then increased in the second hidden layer to the number of channels using sigmoid activations. The outputted thresholds are used as channel-attention weights to scale the channels from the initial convolutional layer, which are multiplied by a residual connection parallel to the Squeeze and Excitation Network, also described as "excitation operation". An improved version of the Squeeze and Excitation Network is the Convolutional Block Attention Module (CBAM) proposed by Woo et al. [45]. CBAM uses two inputs for the Squeeze and Excitation Network, channel-attention and spatial-attention, which are obtained a max and an additionally average pooling operation. This thesis experiments with the CBAM approach. The comparison with this model is motivated by directly evaluating the performance of another attention-based mechanism, such as channel-attention, in the experiments, while considering the model independently compared to [15] and [37], which use the model in combination with other approaches. The modified version for ECG signals in this thesis is based on the CBAM architecture [45] with three (64, 128 and 256) channel-attention blocks. The output is densed using a global average pooling and a feed-forward layer with sigmoid activations to 26 classes.

3.3.6 Proposed approach

The proposed approach is motivated by the idea of applying Multi-head Attention on feature maps obtained from multi-scaled convolutions. Similarly, as illustrated in 3.3.4, it uses 1x3, 1x5, 1x9 and 1x15 kernels to capture small and large sized features. 256 channels, from which each 64 correspond to a different kernel size, are then forwarded into 3 consecutive Transformer encoder blocks to capture temporal relations among the channels. A step that is not shown in the graph is that the tensor of all channels is transposed from (batch, 250, 256) to (batch, 256, 250) before it is fed into the first Transformer encoder block, meaning that instead of treating all channel values in depth as embedding, each individual channel is passed as embedding to the Transformer encoder, thus the depth of all channels represents the sequence. The previous has been analysed, but lead to worse training performances. The worse training performance might be duo that the projected values within each kernel are encapsulated and considered isolated from each kernel, as the embeddings are build from taking only one value within each feature map, but from all feature maps. This approach tries to learn relational features based on isolated information within each kernel, which does not represent the projected kernel information as whole. In other words the network first extracts features from 1x3, 1x5, ... kernels and then the extracted kernel information is destroyed by mixing it up with other kernel values into separated embeddings. Each Transformer encoder block is composed of the same Transformer encoder layers as described in 3.3.2. The Multi-head Attention layer uses 8 heads with $qkv_{dimension}$ 32. The feed-forward hidden layer is 24. The corresponding model is illustrated in 3.7.

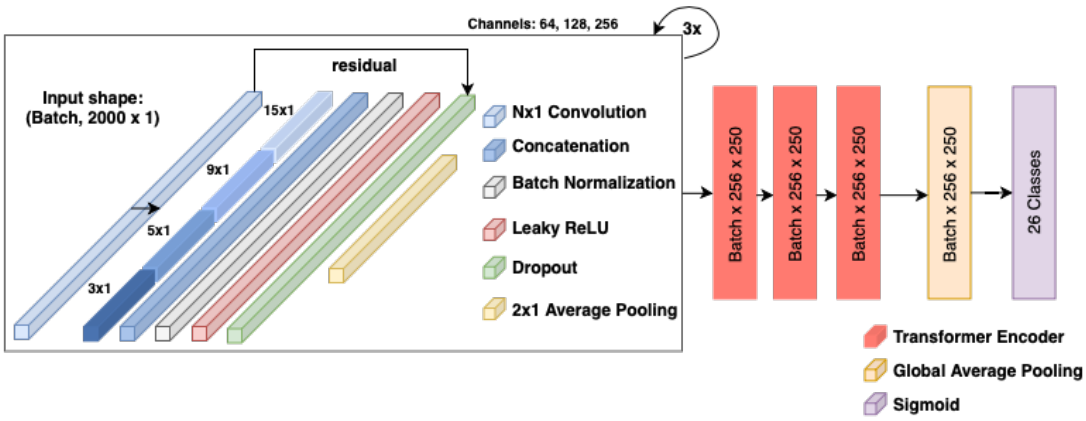

Figure 3.7: Multi-scale residual Convolutional Network combined with three Transformer encoder blocks

The Transformer encoder block does not incorporate positional encoding. The dimensions between each Transformer encoder are equal for stacking. The output of the final Transformer encoder block is fed into a global average pooling layer and a final dense layer times 26 classes, since the multi-label annotated Physionet 2021 challenge focuses on 26 different classes, using sigmoid activations to give for each class a probability score. Previously a flattening layer has been conducted, which did let the model overfit to quickly. Although, the proposed approach has also been implemented and experimented with the Python framework TensorFlow 2, in the appendix A is the entire equivalent implementation of the model given for PyTorch 2. It shows additionally step-by-step the shapes within each network layer (see "Combined model" part).

3.3.7 Multi-lead models and model transfer

3.3.8 Fine-tuning

The Physionet 2021 challenge organisers use a scoring metric, which gives partial rewards to misdiagnoses that result in similar treatments or outcomes than the true diagnosis, as assessed by professional cardiologists. This scoring metric is intended to reflect the clinical reality in which some misdiagnoses would have more negative consequences to a patient's treatment than others, since confusing some classes is less harmful than confusing others (see Physionet 2021 challenge website). The challenge score is calculated by incorporated class evaluation weights for each correct and incorrect classification. To fine-tune the performance of the models on the Physionet 2021 challenge metrics the presented approaches adopt an altered custom lost function used by the first ranked model of the Physionet 2021 challenge [27], which was initially proposed by Vicar et al. [40] [41]. The custom loss function incorporates the evaluation weights of the challenge and decreases the total loss as follows [27]:

$$\text{Loss} = \sum^{batch} \left(\text{BCE}(T, P) - \text{CL}(T, P) + \text{SL}(P) \right) \tag{3.5}$$

$$\text{CL}\left(T, P\right) = \sum_{i=1}^{class\ n} \sum_{j=1}^{class\ n} \omega_{ij}\, a_{ij} \tag{3.6}$$

$$\text{SL}\left(P\right) = \text{average}\left(-4P\left(P-1\right) \right) \tag{3.7}$$

The calculated total loss of the mini-batch is composed of three functions, first the standard binary cross-entropy loss for multi-label classification (BCE), second a custom loss function (CL) and third a sparsity loss function (SL). T and P are 26-dimensional vectors that contain the true labels as binary values and predicted output classes as continuous values. The sparsity loss (SL) function adds a penalty value ranging between 0 to 1 for each outputted logit that is close to 0.5. It is a down concaved parabolic curve with vertex at 0.5 and roots 0 and 1. The sparsity loss is intended to improve the final threshold optimization, as it forces the model to output values close to 0 or 1. The custom loss function incorporates the evaluation weights w_{ij} (see class evaluation weights), where a_{ij} are values obtained from a multi-label confusion matrix A. This confusion matrix is calculated as in (3.7) and (3.8) below, where N is a normalizing constant obtained from the continuous version of the logical OR [40]:

$$A = T^T \left(\frac{P}{N} \right) \tag{3.8}$$

$$N = \left(\left(T + P - T \odot P \right) \mathbf{1}_{c\times 1} \right)\ \mathbf{1}_{1\times c} \tag{3.9}$$

$\mathbf{1}_{c\times 1}$ and $\mathbf{1}_{1\times c}$ are 26-dimensional vectors consisting of only ones, $\frac{P}{N}$ and $\odot$ are element-wise division and multiplication (Hadamard) operators.

The proposed deep-learning approaches in this thesis are fine-tuned with this compound cost function to maximize the challenge score. The fine-tuning is performed on an added small top-layer network, which is concatenated to each base-model after pre-training. The small network consists of one hidden-layer and an output-layer, both with 26 nodes, as the number of outputted classes, and sigmoid activations. Therefore, the deep-learning models are pre-trained and treated as base-models with standard binary cross-entropy loss for multi-label classification

(BCE), trained for at most 100 epochs, to maximize the validation accuracy and then concatenated with the small top-layer network and fine-tuned for 15 epochs to maximize the challenge score. This idea is adopted from [27], which utilize the cost function in combination with a small fine-tuned network, which either does not propagate gradient updates to the base-model. Additionally, a grid search threshold optimization, iterating 1000 decimal places, is performed on the fine-tuned model that optimizes the F1 score for each class independently.

Chapter 4

Evaluation

4.1 Datasets

4.1.1 Physionet 2021 challenge database

This section summarises the publicly available Physionet 2021 challenge database that is used to train all models within this thesis. In section 4.1.2 the MyDiagnostick data provided by the Maastricht University is briefly discussed. The Physionet 2021 challenge database provides 12-lead ECGs recorded with wearable 12-lead Holter monitor systems, which are annotated by cardiac experts. It is a diverse dataset collection from seven sources in four countries and three continents (see Physionet 2021)[31]. The sources of the challenge data include the Chapman-Shaoxing database and Ningbo database, PTB and PTB-XL database, Georgia 12-lead challenge data, CPSC and CPSC-extra database and the INCART database. The challenge data contains 88,253 publicly shared 12-lead ECG recordings in total. The annotations contain labels from more than 100 known arrhythmia types (see all Physionet 2021 database arrhythmias). Each arrhythmia class has a "Snomed CT code" assigned, an unique ID assigned to each arrhythmia type. The ECGs are multi-label annotated, meaning that multiple arrhythmia types can occur within the same ECG. Table 4.1 shows an overview of the Physionet 2021 challenge database composition, here dataset source, recording length in seconds, sampling frequency and ECG samples proportion.

Dataset source	**Recording length in seconds**	**Sampling frequency**	**ECG samples**
Chapman-Shaoxing database & Ningbo database	10s	500 Hz	45,152
PTB database & PTB-XL database	10-120s	either 500 or 1000 Hz	22,353
Georgia 12-lead challenge database	5-10s	500 Hz	10,344
CPSC database & CPSC-extra database	6-144s	500 Hz	10,330
INCART 12-lead database	1800s (30 min)	257 Hz	74

Table 4.1: Physionet 2021 challenge data composition

This thesis only uses the published training data for training and evaluation, since the test data for the official challenge and evaluation scoring metrics are withhold by the organisers and are not publicly available. However, given the information on the official Physionet 2021 challenge website that 88,253 ECGs are publicly shared for training and 36,266 ECGs are retained privately as test data, the assumption for the experiments 4.4 is that the label distribution in the private test set is similar to the public training data 4.1 (bottom graph). Following this assumption and using the challenge scoring metrics 4.4 for a benchmark of the models with other model results within the challenge, will make the experiments 4.4 to some extend comparable. Figure 4.1 shows the class distribution for the entire publicly available Physionet 2021 database (top), as well as the distribution of the scored classes within the challenge (bottom). The official Physionet 2021 challenge uses a subset of 30 classes that are scored within the challenge (see scored Physionet 2021 database arrhythmias). Due to limited samples, eight arrhythmia types are treated as same class pairs within the challenge metrics, namely Premature Atrial Contractions (PAC) and Supraventricular Premature Beats (SVPB), Premature Ventricular Contractions (PVC) and Ventricular Premature Beats (VPB), Complete Left Bundle Branch Block (CLBBB) and Left Bundle Branch Block (LBBB), Complete Right Bundle Branch Block (CRBBB) and Right Bundle Branch Block (RBBB). Therefore, to benchmark the proposed approaches, the developed models follow these requirements and annotations are similarly grouped in the training data, yielding 26 distinct class annotations instead of 30, while other class annotations and ECGs samples that are not part of the scored metrics are discarded. Both graphs in 4.1 show that the data is quite unbalanced. Considering the bottom graph that show the scored challenge data distribution, 28,971 (22.40%) Sinus Rhythm samples are provided, while many classes with less than 2,000 (1% <) samples are present. For an unbiased model training the dataset needs to be pre-processed and properly balanced to avoid overfitting on some major classes, which is described in section 4.1.3.

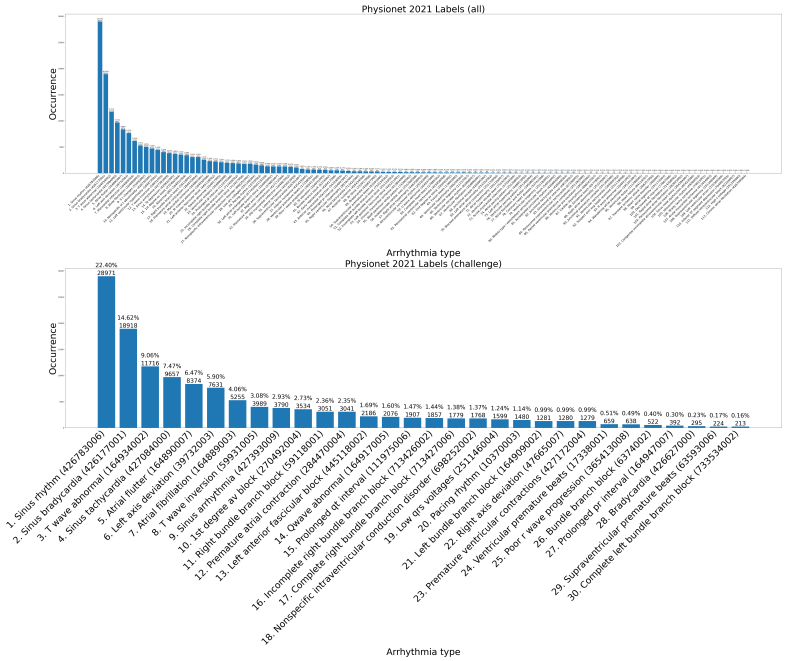

Figure 4.1: Physionet 2021 classes distribution

4.1.2 MyDiagnostick data

The MyDiagnostick data contains single-lead ECGs with a sampling frequency of 200 Hz and 60 seconds in recording length. The MyDiagnostick ECGs were recorded by a special portable device that uses a stick that is hold against the patient's chest instead of placing electrodes. More information about the device can be found here [39]. The obtained recordings are similar to the first lead (lead I) recorded by a 12-lead Holter monitor system. For the evaluation of the transferred models on the MyDiagnostick data, part of the experiments will focus only on a subset of 5 classes, namely Sinus Rhythm (SR), Atrial Fibrillation (AF), Atrial Flutter (AFL), Premature Atrial Contractions (PAC) and Premature Ventricular Contractions (PVC). This is due to the limited class annotations in the MyDiagnostick data, which are limited to these class annotations. ECG pre-processing steps to transfer the models are discussed in the next section 4.1.3. Figure 4.2 shows the class distribution of the corresponding classes for both, the Physionet 2021 data and the MyDiagnostick data. The most underrepresented class in both datasets is Premature Ventricular Contraction with 1279 samples and 160 samples, respectively. In many MyDiagnostick samples both classes, Atrial Ventricular Contractions and Premature Ventricular Contractions, are present within the same ECG. Although, the MyDiagnostick provides 10,587 manually annotations, the dataset consists of more than 40,000 samples. The remaining samples are automatically labelled as either Sinus Rhythm or Atrial Fibrillation by an assistive ECG device. As part of this thesis the aim of the experiments will be to evaluate the best performing pre-trained model from the Physionet 2021 challenge data on the manually annotated MyDiagnostick data, which will be discussed in section 4.2.2, and to provide annotations for the remaining 30,000 samples.

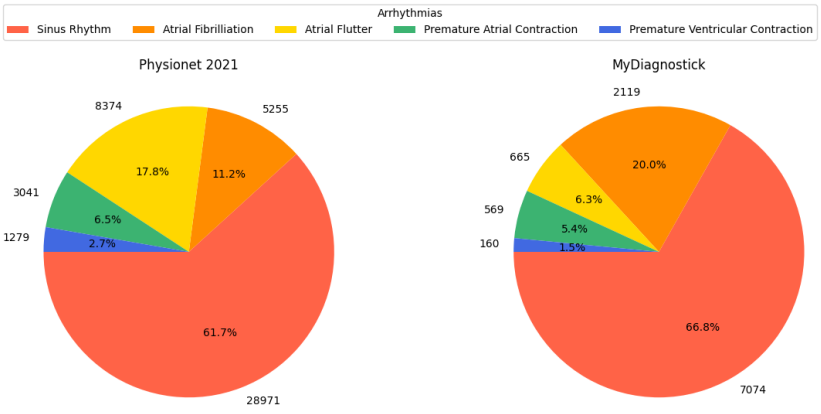

Figure 4.2: Physionet 2021 and MyDiagnostick class distribution - SR, AF, AFL, PAC and PVC

4.1.3 Pre-processing

This section summarises the pre-processing steps that have been applied on both datasets to address data inequalities. Pre-processing steps include train-test split by patients, class balancing, zero-padding/truncation, resampling, normalisation and filtering. The train-test splits are: 50% training, 25% validation and 25% test set. This choice has been made, because the dataset is to

large to evaluate all models using cross-validation, so there is a sufficient amount of data in the validation and test sets. The Physionet 2021 challenge data was previously split by patient to ensure that the models learn general features rather than specific features of individual patients. The MyDiagnostick data is only utilized for transferring and testing the pre-trained models from the Physionet 2021 challenge data, meaning the MyDiagnostick data is not used for fine-tuning. As in figure 4.1.3 shown, the training data is highly unbalanced. To address this issue, standard class down- and upsampling is applied on the Physionet data using the Python package imbalanced-learn. In the case of the 26-class models down- and upsampling is set to 2000 samples by randomly discarding or duplicating some ECGs. In the case of the 5-class models down- and upsampling is set to 6000 samples. This ensures that the minor classes are not oversampled to often, because in the 26-class dataset the most underrepresented class has about 500 samples and in the 5-class dataset about 1200 samples. The class down- and upsampling was applied on the data after splitting the data by patients into train and test sets to avoid evaluating on already trained ECGs. In the test data only downsampling is applied. The validation set represents a random sampled subset from the training set. However, since the data is multi-label annotated the class balance is only to some extend achieved, because imbalanced-learn does not support class balancing for multi-label classification. The workaround solution was to transform the multi-label annotations into multi-classification annotations and balance it from there. This means that the data samples are transformed from multi-hot encoded labels, e.g. "[0, 0, 1, 0, 1]", into two duplicated examples with categorical labels, e.g. 2 and 4 in this example. The sample are then splitted into separate bins by classes and ascending sorted from minority to majority classes. From here, each underrepresented bin has been then upsampled to 2000 or 6000 samples, respectively. Next, ascending from minority to majority classes the ECGs within each bin are added successively to the balanced datasets using the original multi-label annotations, while for the majority classes with samples above the upsampling thresholds duplicates are ignored using the patient IDs and added until 2000 or 6000 samples have been reached. This ensures to add as many different ECGs to the balanced dataset as possible, since the majority classes provide enough samples so that duplicate annotations containing one of the minority classes can be ignored. The class balancing results are shown in figure 4.1.3. The second graph on the left for the 26 balanced train set classes shows still unbalanced data, because many samples from the minority classes contain more than one label with one of the frequent classes, e.g. Sinus Rhythm. This could be addressed by further downsampling the majority classes, e.g. majority classes with only one annotation could be removed, but this was not done in favour of having more available ECG samples in each set.

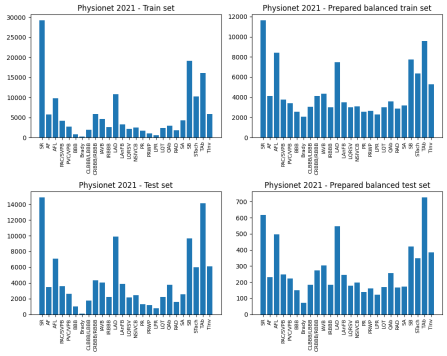

Figure 4.3: Balanced Physionet 2021 data (All scored classes)

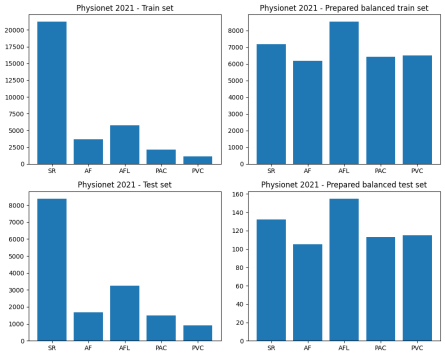

Figure 4.4: Balanced Physionet 2021 data (5 classes)

Since the Physionet 2021 challenge data consists of 12-lead ECGs and the MyDiagnostick data consists of single-lead ECGs, all models in the experiments are pre-trained on the first lead (lead I) from the Physionet data, which is the most similar lead to the first lead from a 12-lead Holter monitor system. Most of the ECGs in the Physionet data are recorded at a sampling frequency of 500 Hz, while the MyDiagnostick ECGs are recorded at a sampling frequency of 200 Hz. Most of the Physionet 2021 challenge ECGs contain 10 second long recordings (5000 data points for 10 seconds - see 4.1). In comparison the MyDiagnostick data contains 60 seconds long recorded ECGs with 12000 data points (2000 data points are equal to 10 seconds). To address the data inequality, first each ECG in the Physionet 2021 challenge is truncated or zero-padded to 5000 data points. Next, each Physionet ECG is downsampled to 200 Hz, yielding 2000 data points. This makes both datasets uniform in terms of sample density and temporal resolution. In addition, each ECG is normalised within the range of -1 and 1 using min-max scaling. This is particularly helpful in reducing signal amplitude variation due to differences in electrode placement, body size and individual heart activity. In addition, normalisation helps to stabilize training with gradient descent and avoids gradient explosion problems during backpropagation. By keeping the model weights in an uniform range prevents the model from being biased towards certain input features (e.g. R peaks). In the next step, a standard butterworth bandpass filter is applied with a bandwidth of 0.3 Hz to 21 Hz, which reduces small noise fluctuations in the ECG signal and facilitates the model to learn key features from arrhythmias, such as RR intervals. Finally, all ECGs in the Physionet data are again zero-padded to 12000 datapoints. This is necessary, because a model applied on both datasets need to receive as input an equal sized feature vector. Figure 4.5 visualizes step by step the ECG pre-processing procedure. The last step after the filtering is only utilized for the model that is transferred and tested on the MyDiagnostick data. The last is an ECG example from the MyDiagnostick data to show the mismatch between the lengths of the ECGs, which only needed to be normalised and filtered as it is already present with a sample frequency of 200 Hz.

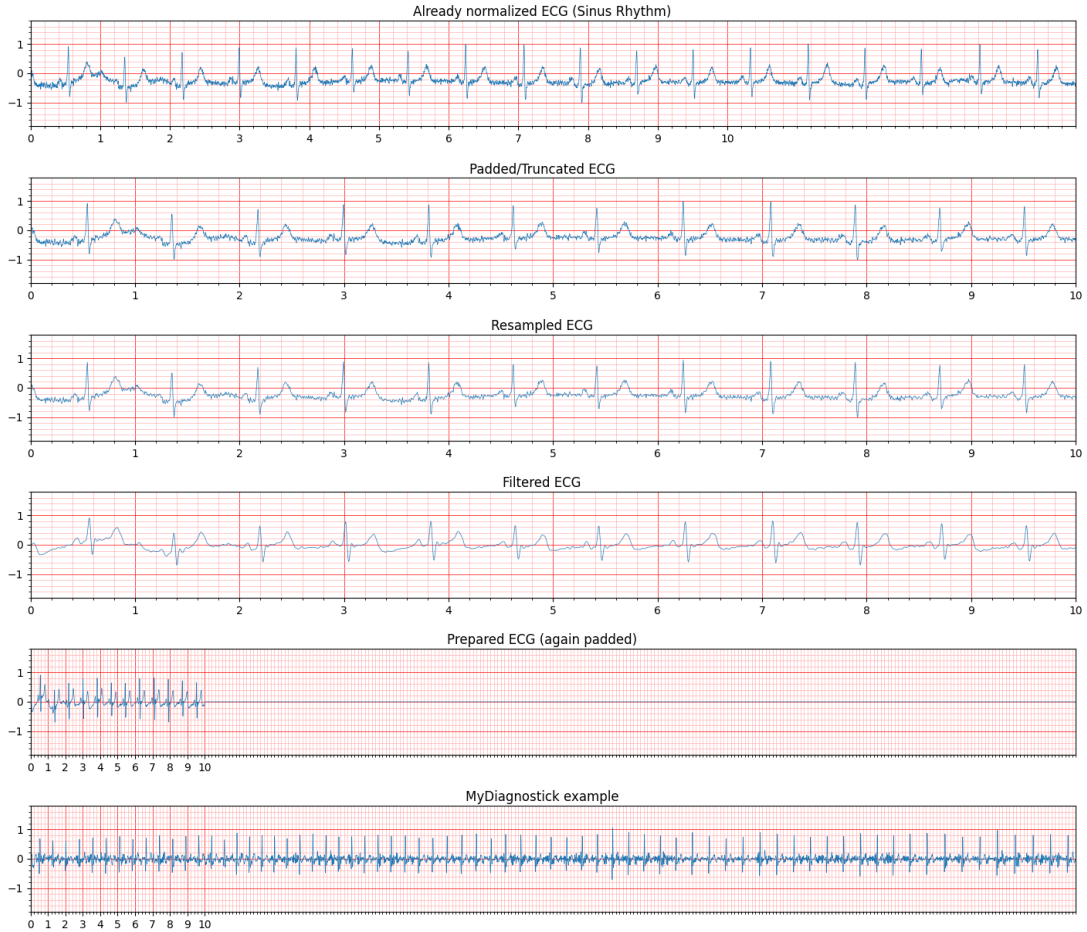

Figure 4.5: ECG pre-processing steps applied to the Physionet 2021 data

4.2 Experimental setup and results

This section describes the experimental setup and results of the developed approaches. This section only describes the experimental setup and results and highlights some observations, while section 5 discusses in detail the results. The section is divided into two parts. The first part describes the experimental setup and presents the results for the models trained and tested on the Physionet 2021 database. The second part describes the experimental setup and presents the results of the pre-trained models on the Physionet 2021 database transferred and evaluated on the MyDiagnostick data. All models have been trained in Google Colab using a Nvidia T4 GPU with 16GB VRAM for at most 100 epochs and early stopping enabled with regard to not decreasing validation loss (patience set to 5). Adam optimizer is used with an initial learning rate of 0.001, which decreases by 1/10 with regard to not decreasing validation loss (patience set to 5). As loss function binary-crossentropy for multi-label classification is used, because the Physionet 2021 and MyDiagnostick data are both multi-label annotated. All models use sigmoid activations in the final output layer times the number of classes. For simplicity, all models are trained with a fixed mini-batch size of 32, although this could also be determined empirically. Due to minimize

computational costs and required training time, as some developed models took up to one hour for training, cross-validation is not applied. However, cross-validation could evaluate the generalization ability of the models better. The experiments focus on the fixed train-test split, which was previously described 4.1.3, and use the balanced training data. For testing the unbalanced test set is used. This is due the assumption that the official Physionet 2021 challenge test set, which is not available, is also unbalanced. The train-test samples within these experiments are identically for each tested model. Using scikit-learn the weighted average of accuracy, precision, recall and F1 is computed for each model. In addition, the official Physionet 2021 evaluation metrics are calculated to benchmark the developed approaches with other approaches within the challenge. For other model results see challenge results. The official Physionet challenge metrics use for evaluation AUROC, AUPRC, Accuracy, F1 measure and the calculated challenge score, which is based on the Physionet 2021 challenge class evaluation weights.

4.2.1 Physionet 2021 challenge

The analyses in this section address the first two research questions of this thesis based on the Physionet 2021 challenge data:

1. How well does a Transformer-based model perform on the Physionet 2021 challenge data compared to a feature-based model or a Convolutional Network?

2. Can an ensemble model of Transformer and Convolutional Network effectively capture spatio-temporal information and improve accuracy?

As part of the experiments to address the first research question, it will be investigated whether a Transformer-based model alone without any modifications is able to extract features from ECGs. For this several variations of a Transformer encoder 3.3.2 without extracted features as input, e.g. convolutional features, are trained on the ECGs from the Physionet 2021 challenge for the classification of 26 arrhythmias. These experiments are motivated by evaluating the Transformer encoder and Multi-head Attention mechanism applied directly on the ECG signals without feature extraction and whether some good hyperparameter configurations of the model can be found. Gridsearch has been applied on various parameters. Investigated parameters include input shape, positional encoding, number of stacked encoder blocks, number of heads, dimensions of the projection matrices q, k and v, feed-forward layer dimension and dropout rate. The input shape describes in how many embeddings the inputted ECG sequence is splitted before fed into the Transformer encoder block. (40, 50) means that the pre-processed ECG with 2000 datapoints in length is splitted into 40 consecutive segments (embeddings), where each embedding contains 50 datapoints. Positional encoding, which is optional, adds temporal information before the Transformer encoder block to each embedding, based on the formula described in 3.1.1. Number of heads and q, k and v dimension describe the size of the q, k and v matrices that project the embeddings into the attention feature space and parallel attention blocks in each Transformer encoder layer, as described in 3.1.3. Feed-forward dimension determines the number of nodes in the hidden-layer of the feed-forward layer that is part of each encoder block. The amount of trainable parameters of the encoder block is primirarly determined by the above three parameters. Dropout is applied to all layers, either set to 0.1 or 0.4. The corresponding results are shown in table 4.2. Based on the experimental results it can be seen that a standard Transformer encoder-based model is able to extract relational features from the ECGs for the classification of 26 different arrhythmias. However, in terms of accuracy there is still room for improvements, while precision is higher. Using smaller ECG segments as input sequence, e.g. (40, 50) instead of (10, 200), offers slightly improved performance. Surprising is that positional

encoding negatively affects performance. Based on the results the size of the model and amount of trainable parameters, e.g. number of encoder blocks, q, k and v dimensions and feed-forward layer size, improves the model performance, but which should be critically adopted in terms of computational costs, especially considering a model with around 70,000,000 parameters, which requires much more computional resources and time to train. The effect of a higher dropout rate varies among the experiments, although in the last model it slightly improves the generalization ability of the model.

Input shape	**Pos. enc.**	**Enc. blocks**	**Heads**	**qkv dim**	**ff dim**	**Drop-out**	**Train. param.**	**Acc.**	**Prec.**	**Rec.**	**F1**
(40, 50)	True	1	1	25	24	0.1	155.375	0.096	0.514	0.120	0.194
(40, 50)	True	1	1	25	24	0.4	155.375	0.065	0.418	0.083	0.139
(40, 50)	True	8	1	25	24	0.1	878.818	0.061	0.579	0.079	0.139
(40, 50)	True	8	1	25	24	0.4	878.818	0.098	0.592	0.122	0.203
(40, 50)	False	1	1	25	24	0.1	155.375	0.227	0.747	0.25	0.374
(40, 50)	False	1	1	25	24	0.4	155.375	0.224	0.742	0.253	0.378
(40, 50)	False	8	1	25	24	0.1	878.818	0.228	**0.765**	0.261	0.389
(40, 50)	False	8	1	25	24	0.4	878.818	0.226	0.737	0.255	0.379
(10, 200)	False	8	1	25	24	0.1	1.004.818	0.160	0.744	0.174	0.283
(10, 200)	False	8	8	25	24	0.1	2.129.018	0.177	0.709	0.197	0.308
(40, 50)	False	8	8	400	24	0.1	6.035.018	0.223	0.762	0.247	0.374
(40, 50)	False	8	8	25	2048	0.1	65.947.210	0.219	0.740	0.257	0.382
(40, 50)	False	8	8	400	2048	0.1	70.819.210	0.226	0.751	0.257	0.383
(40, 50)	False	8	8	400	2048	0.4	70.819.210	**0.245**	0.739	**0.286**	**0.413**

Table 4.2: Standard Transformer encoder evaluated on the Physionet 2021 challenge data

The next experiments compare all developed models 3.3 with the proposed approach 3.3.6 ("Multi-scale CNN and Transformer encoder" in 4.3 and 4.4) trained on the ECGs from the Physionet 2021 challenge for the classification of 26 arrhythmias. 4.3 and 4.4 show the corresponding experimental results after each model has been fine-tuned and optimized by thresholds 3.3.8. Table 4.4 shows the results based on the evaluation metrics from the Physionet 2021 challenge. Table 4.3 has two values for accuracy, while the left accuracy correspond to the accuracy score before the models have been fine-tuned and threshold optimized 3.3.8. These accuracy scores are higher, because the models optimized on the challenge evaluation metrics also prefer false positive classifications due to the incorporated partial added rewards for misclassifications, thus the overall accuracy decreases. An investigated constraint within these experiments are the model sizes 4.3. Given the second results on the challenge metrics experiments 4.4, all models are able to achieve high performances compared to the random forest, which has not been additionally fine-tuned on the challenge metrics with a small top-layer network. Most remarkable in the experiments is that the feature-based random forest yields the highest accuracy, closely followed by the residual Convolutional Network and the proposed approach. In terms of precision, recall and harmonic F-measure the random forest classifier and the residual Convolutional Network show in both evaluation tables highest performance. The Transformer encoder ("Transformer encoder with 8 heads and blocks") without any feature preparation performs noticeably lower than the proposed improved version ("Multi-scale CNN and Transformer encoder"), which uses extracted features from multi-scale convolutions as input for multiple stacked Transformer encoder blocks. Moreover, the model is able to perform better than the modified versions of the Squeeze and Excitation Network and the multi-scale Convolutional Network. Beside, the

influence of dilated convolutions within the residual Convolutional Network does not affect the model performance. Among all investigated models the Squeeze and Excitation Network has the fewest parameters, while it also shows lower performance. A good trade-off between model size and performance offers the residual Convolutional Network with 3,702,282 and the random forest with 5,556,212 parameters.

Model	Parameters	Accuracy	Precision	Recall	F-measure
Random Forest (Biobss features)	5.556.212	**0.390**; /	**0.714**	0.382	0.406
Residual CNN without dilation	3.702.282	0.387; 0.310	0.593	0.692	**0.625**
Residual CNN with dilation	3.702.282	0.386; 0.311	0.590	**0.695**	0.624
Transformer encoder (8 heads, 8 blocks)	2,129,018	0.225; 0.100	0.380	0.619	0.432
Multi-scale Convolutional Network	53.202.522	0.358; 0.299	0.581	0.673	0.609
Modified Squeeze and Excitation Network	**152.210**	0.257; 0.184	0.463	0.622	0.501
Multi-scale CNN and Transformer encoder	11,105,058	0.371; 0.308	0.579	0.689	0.616

Table 4.3: Physionet 2021 train/test split evaluation

Model	AUROC	AUPRC	Accuracy	F-measure	Challenge metric
Random Forest (Biobss features)	0.560	0.142	**0.390**	0.153	0.147
Residual CNN without dilation	**0.874**	**0.439**	0.310	**0.482**	**0.556**
Residual CNN with dilation	0.842	0.404	0.311	0.442	0.554
Transformer encoder (8 heads, 8 blocks)	0.701	0.211	0.100	0.258	0.402
Multi-scale Convolutional Network	0.832	0.382	0.299	0.414	0.526
Modified Squeeze and Excitation Network	0.768	0.290	0.184	0.322	0.445
Multi-scale CNN and Transformer encoder	0.843	0.403	0.309	0.446	0.547

Table 4.4: Physionet 2021 challenge metric scores evaluation

The next experiments focus primarily on the second research question, whether the proposed combination of Transformer model and a Convolutional Network can capture spatio-temporal features from multi-lead ECGs. The corresponding results are shown in table 4.5 and 4.6.

Model	Parameters	Accuracy	Precision	Recall	F-measure
Multi-scale CNN and Transformer encoder (2-leads)	2.676.706	0.352; 0.197	0.537	0.691	0.582
Multi-scale CNN (2-leads)	**1.808.794**	**0.362**; 0.216	0.555	0.716	0.598
Multi-scale CNN and Transformer encoder (6-leads)	6.444.770	0.353; 0.250	0.556	0.651	0.584
Multi-scale CNN (6-leads)	5.426.330	0.350; 0.213	0.549	0.697	0.584
Multi-scale CNN and Transformer encoder (12-leads)	12.096.866	0.357; **0.269**	**0.562**	**0.714**	**0.604**
Multi-scale CNN (12-leads)	10.852.634	0.351; 0.185	0.543	0.709	0.595

Table 4.5: Physionet 2021 train/test split evaluation for multiple leads

Model	AUROC	AUPRC	Accuracy	F-measure	Challenge metric
Multi-scale CNN and Transformer encoder (2-leads)	0.770	0.336	0.197	0.379	0.510
Multi-scale CNN (2-leads)	0.802	0.362	0.216	0.404	0.527
Multi-scale CNN and Transformer encoder (6-leads)	0.797	0.348	0.250	0.392	0.487
Multi-scale CNN (6-leads)	**0.805**	0.346	0.213	0.383	0.495
Multi-scale CNN and Transformer encoder (12-leads)	0.799	**0.367**	**0.269**	**0.409**	**0.552**
Multi-scale CNN (12-leads)	0.795	0.351	0.185	0.386	0.509

Table 4.6: Physionet 2021 challenge metric scores evaluation for multiple leads

The experiments show three setups in which the performance of the proposed approach modified for multiple leads, described in section 3.3.7, compared with the multi-scale Convolutional Network is evaluated using 2-lead (lead I and II), 6-lead (I, II, III, aVR, aVL and aVF) and 12-lead (all leads provided) ECGs from the Physionet 2021 data. Surprising is that the results show that the models trained on more leads do not improve the overall performance a lot compared with the models trained on fewer leads. Moreover, the performance between the two investigated models are quite similar across the three (2-, 6- and 12-lead) experiments. The 2-lead multi-scale Convolutional Network shows the highest accuracy before fine-tuning 4.5, while using the fewest parameters. In comparison, does the proposed multi-scale Convolutional Network with three Transformer encoder blocks extended to 12-leads shows the highest performance on all other metrics, including the challenge metrics, while it needs the most parameters within these experiments. A detailed discussion about the observed results is given in section 5.

4.2.2 MyDiagnostick

The analyses in this section address the last two research questions of this thesis based on the Physionet 2021 challenge and MyDiagnostick data:

1. How is the performance of the most promising deep-learning model at discriminating SR, AF, AFL, PAC and PVC on both datasets?

2. What are the challenges in transferring the pre-trained models from the Physionet 2021 challenge data to the MyDiagnostick database? Do the models generalise well, even though different ECG devices were used?

In table 4.7 and 4.8 are the corresponding results for the best performing model, here the residual Convolutional Network without dilation, pre-trained on a subset of the Physionet 2021 challenge data with the five investigated classes. Moreover, figure 4.6 shows a modified confusion matrix for multi-label classification. As shown in the results below the model has difficulties with predictions in the MyDiagnostick database. Base on the confusion matrix 4.6 the model tends to label all instances as Atrial Fibrillation and partially Sinus Rhythm. By revisiting 4.5 the reason could be that the model fails due to the high inequalities between the recording lengths. The model has difficulties in learning from mainly zero-padded data and then being transferred and tested on ECGs with a larger recording length. Another reason could be that the data is not appropriate filtered or resampled. However, based on the observations in 4.7 and the left confusion matrix in 4.6, the model performs acceptable within the Physionet 2021 data. Previously, the idea was to address this issue by splitting each MyDiagnostick ECG into six segments with 10 seconds in length and classify each segment individually and add up the probabilities. The Problem that arises here would be that some arrhythmias, e.g. Premature Atrial Contractions or Premature Ventricular Contractions, may occur only once in an ECG. The summed probabilities may shrink the models overall outputted probabilities to classify an ECG as such. Further considerations could be to test other approaches, e.g. the proposed multi-scale Convolutional Network with several Transformer encoder blocks. Based on the left confusion matrix 4.6 a typical pattern can be seen, which is the confusion between Atrial Fibrillation and Atrial Flutter. Related work [33] [42] highlight these issues. In a prior experiment it has been conducted, which is not shown in this thesis, whether incorporating penalty weights into the binary-crossentropy loss function for misclassifications between Atrial Fibrillation and Atrial Flutter could probably improve the classification accuracy between those two. The experiments turned out to destabilize the training and prevent the models from learning. Table 4.8 shows in addition the evaluation results of the 26-class model. These are lower, because the model is trained on more classes, so misclassifications are more likely.

Model	Accuracy	Precision	Recall	F-measure
Residual CNN without dilation (5 classes)	0.797	0.859	0.814	0.828

Table 4.7: Results of the residual CNN on the Physionet 2021 challenge data (5 classes)

Model	Accuracy	Precision	Recall	F-measure
Residual CNN without dilation (5 classes)	0.402	0.744	0.414	0.432
Residual CNN without dilation (26 classes)	0.111	0.899	0.210	0.240

Table 4.8: Results of the residual CNN on the MyDiagnostick database (5 and 26 classes)

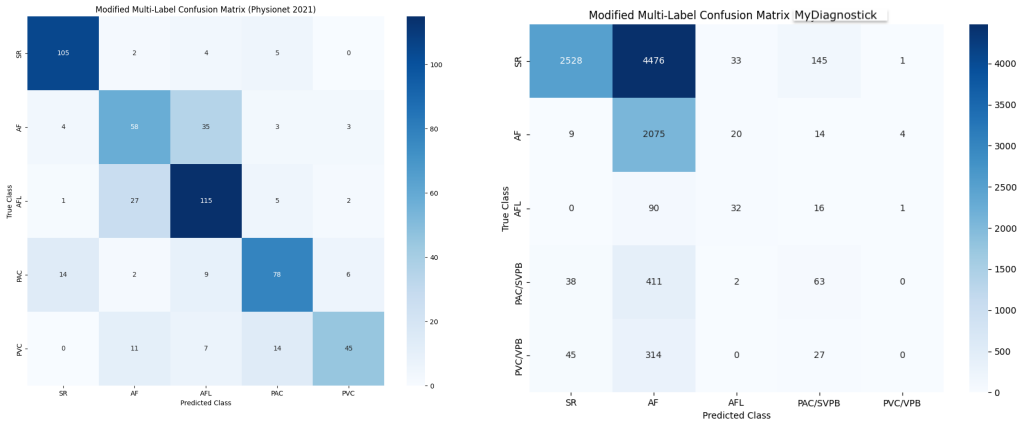

Figure 4.6: Results of the 5-class residual CNN model tested on the Physionet 2021 data (left) and transferred and tested on the MyDiagnostick data (right)

Chapter 5

Discussion and Conclusions

5.1 Discussion

Based on the experiments on the Physionet 2021 challenge data 4.3 4.4 it can be concluded that the proposed approach, the multi-scale Convolutional Network combined with three stacked Transformer encoder blocks, shows improved performances compared with other convolution- and attention-based deep-learning approaches, such as the Transformer encoder model without convolutional features, the multi-scale Convolutional Network or the Squeeze and Excitation Network architecture modified for ECG classification. However, the residual Convolutional Network performs best among all deep-learning models within the Physionet 2021 challenge data experiments 4.3 4.4. The proposed model, which uses convolutional features as input to the Transformer encoder blocks, performs better compared to a Transformer encoder model that does not utilize convolutional features as input and applies the Multi-head Attention mechanism directly on the ECG signals, respectively fixed-size embedding segments from the signal 4.2. Comparing the standalone Transformer model for ECG classification cross-domain with a Vision Transformer [10] for image classification, which shows good performance in image classification tasks and also does not use convolutional features as input, raises the question of what could be the main reason for the performance gap across the domains using a similar approach. As these applications can not be compared, the quality of the data or the complexity of the classification task, here 26 arrhythmia classes within the Physionet 2021 challenge data, could be the main reason for unexpected weak performance. Another possible reason may be related to the characteristics of ECG signals. Depending on the image classification task, the variance of pixels in image data is higher compared to wave segments in ECG signals, which are periodic and repetitive. This condition requires the Transformer encoder model to learn fine-grained and distinct patterns from the signals, which becomes even more difficult for similar arrhythmia patterns across different classes. Following this assumption, it could be that the arbitrary locations of the waves among the inputted ECG segment embeddings confuse or bias the Multi-head Attention mechanism from extracting fine-granular features and relations among the embeddings. Even adding position information to the embedding segments further degrades the performance of the Transformer encoder model 4.2. One idea, apart from using convolutional features as input, might be to evenly align the waves within each embedding before feeding them to a Multi-head Attention block. A similar approach is taken in the work of Choi et al. [6] with ECGBERT, where the authors first cluster the ECG wave segments and then assign the segments to predefined tokens that represent wave clusters.

However, the combination of a Transformer encoder and a Convolutional Network can extract spatio-temporal features. More analysis on optimizing the most suitable combination of filter sizes and Transformer encoder dimensions could be carried out. Based on the results from the Physionet 2021 challenge metrics All models are able to offer high results within the Physionet 2021 challenge by utilizing the adopted custom loss function [27] [40] [41]. The results can be compared with other model results on the official Physionet 2021 challenge results page. However, the scores are not directly comparable, since the official test set uses another test set than in this thesis. In terms of generalization across datasets, e.g. here investigated on the MyDiagnostick dataset, there are still several challenges in dealing with data inequalities. Based on the performance within the Physionet 2021 challenge, which is re is not representative compared to general weak performance. Based on the upper bound of 40 % accuracy among most models there is still a large performance gap in accuracy. From all models the residual Convolutional Network offers the highest harmonic F-measure. Depending on the application, assuming such a classification model would be remotely integrated into wearable medical devices, it would cause to many false positive and negative classifications. In that sense precision and recall should both be optimized. Deep-learning approaches need generally more parameters, while the proposed approach uses more trainable parameters than the residual Convolutional Network, it still shows compared to a multi-scale Convolutional Network alone improved results. Therefore it might be concluded that the Multi-head Attention mechanism adds performance improvement to the network. This assumption is although refuted, because the residual Convolutional Network still performs best. An idea to improve the proposed model could be to incorporate 1x1 convolutions within the network to reduce the model size. However, simple feature-based classifiers for complex ECG classification tasks, such as a random forest, are able to outperform deep-learning models in terms of accuracy. Additionally, these models are much less computational expensive. Figure 5.1 shows two confusion matrices plots from the residual Convolutional Network based on different training and tests on the Physionet 2021 data. As seen varies the inter-class performance of the model across several runs. This could be utilized for building ensemble models that use majority votes to improve the overall classification performance.

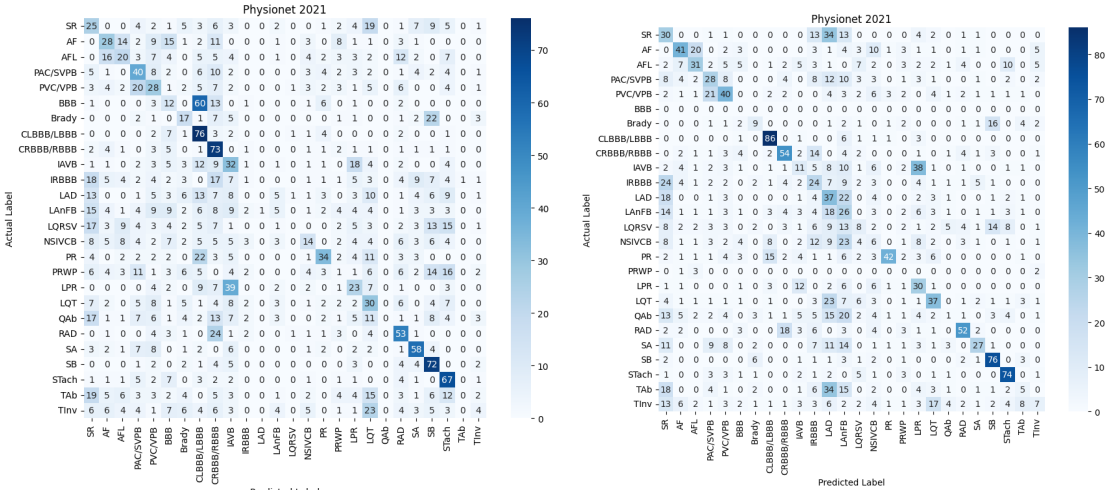

Figure 5.1: Confusion matrices from a residual Convolutional Network based on different training and tests on the Physionet 2021 data

5.2 Conclusions

Based on the experiments and findings the research questions can be answered as follows:

1. How well does a Transformer-based model perform on the Physionet 2021 challenge data compared to a feature-based model or a Convolutional Network?

   Answer: Based on the experimental findings a standard Transformer encoder model without any modifications does not offer good results on extracting meaningful relations and features within ECG signals for classification tasks. Even by adapting the segment sizes or adding positional encoded information further decreases the performances. However, as seen in table 4.2, the model shows slightly improved performances using more parameters. In comparison a Convolutional Network and a feature-based model perform better. The highest accuracy offers the feature-based model, while within the Physionet 2021 challenge the Convolutional Network performs best. The reason that the Multi-head Attention mechanism is not able to capture meaningful features from raw ECGs, might have several reasons. One possible reason, e.g. compared to Vision Transformer [10] that works without convolutions, might be that ECGs contain temporal signals, while images contain spatial features. Many studies that apply Transformer models to ECG classifications [50] [6] [27] [47], do not justify the decisions in details, but reason that the combination works well. An assumption might be that the high variance within each ECG segment, could potentially prevent the Multi-head Attention mechanism without any feature preparation from extracting meaningful relations, as each heartbeat is frequently but randomly placed in each segment.

2. Can an ensemble model of Transformer and Convolutional Network effectively capture spatio-temporal information and improve accuracy?

   Answer: Yes, by feeding feature maps into the encoder block, as proposed in this thesis, improves the model performances as shown in table 4.3 and 4.4. This concludes based on the own findings that the Transformer model is able to capture spatio-temporal information. However, compared to other approaches it does not significantly outperform a Convolutional Network. Based on some prior experiments on tuning the convolutional kernel and layer sizes the Transformer performance varies, which could be further empirically improved by experimenting on different datasets and with different architectures.

3. How is the performance of the most promising deep-learning model at discriminating SR, AF, AFL, PAC and PVC on both datasets?

   Answer: The model shows acceptable but improvable performance within the Physionet 2021 data. However, based on the MyDiagnostick test results the predictions seem to be randomized, which might be due to high signal inequalities. In terms of addressing confusion between Atrial Fibrillation and Atrial Flutter or Premature Atrial Contractions and Premature Ventricular Contractions, could potentially mostly improve such models.

4. What are the challenges in transferring the pre-trained models from the Physionet 2021 challenge data to the MyDiagnostick database? Do the models generalise well, even though different ECG devices were used?

   Answer: The largest challenge arises due to inequalities in recording lengths. Furthermore, the data needs to be equally pre-processed, e.g. same normalisation, resampling

and filtering steps should be applied, thus the ECGs are homogeneous present and equal quality. Although, this thesis tries to address this issues, the incompatibility of the model is still influenced by the large recording length inequality. Another possible solution could be to try further models, e.g. the own proposed multi-channel Transformer model.

5.3 Summary and Outlook

This thesis investigated the application of various multi-scale convolutional and attention-based deep-learning networks for the classification of 26 cardiac arrhythmias. It proposes a novel approach that uses a deep residual Convolutional Network with multi-scale kernels in combination with several stacked Transformer encoder blocks that shows improved performances on the Physionet 2021 challenge compared to other approaches. However, ECG feature extraction for deep-learning models is not a trivial task and need to be revisited carefully to obtain desired results. This includes the generalization ability of these models across diverse ECG datasets, recorded with different monitoring systems. An interesting future direction for complex and accurate multi-label ECG classification tasks in combination with attention mechanisms may focus on experimenting with improved ECG signal data preparation techniques or analyse various feature extraction methods, such as attention-based models in combination with selected filtering techniques or by examining the ability of these models by incorporating frequency domain features, such as the wavelet transformation. In contrast to this research direction suggestion are much less computational expensive feature-based models by using hand-crafted features, which need much fewer time and resources to train. Due to the rapid pace of computational requirements of deep-learning models and environmental impact, improving on sophisticated features should be considered as preferred research direction rather than experimenting with more complex models, such as Transformer models.

Bibliography

[1] Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin. Attention is all you need, 2023.

[2] J. Bender, K. Russell, L. Rosenfeld, and S. Chaudry. *Oxford American Handbook of Cardiology*. 2010.

[3] Pingping Bing, Yang Liu, Wei Liu, Jun Zhou, and Lemei Zhu. Electrocardiogram classification using tsst-based spectrogram and convit. 2022.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[5] Chao Che, Peiliang Zhang, Min Zhu, Yue Qu, and Bo Jin. Constrained transformer network for ecg signal processing and arrhythmia classification. 2021.

[6] Seokmin Choi, Sajad Mousavi, Phillip Si, Haben G. Yhdego, Fatemeh Khadem, and Fatemeh Afghah. Ecgbert: Understanding hidden language of ecgs with self-supervised representation learning, 2023.

[7] Fabiola De Marco, Filomena Ferrucci, Michele Risi, and Genoveffa Tortora. Classification of qrs complexes to detect premature ventricular contraction using machine learning techniques. 2022.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[9] Yanfang Dong, Miao Zhang, Lishen Qiu, Lirong Wang, and Yong Yu. An arrhythmia classification model based on vision transformer with deformable attention. 2023.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2020.

[11] Mohamed Elgendi, Mirjam Jonkman, and Friso De Boer. Frequency bands effects on qrs detection. 2010.

[12] Jeffrey L. Elman. Finding structure in time. 1990.

[13] Ziti Fariha, Ryojun Ikeura, and Soichiro Hayakawa. Arrhythmia detection using mit-bih dataset: A review. 2018.

[14] P. Hamilton. Open source ecg analysis. 2002.

[15] Hyeongrok Han, Seongjae Park, Seonwoo Min, Hyun-Soo Choi, Eunji Kim, Hyunki Kim, Sangha Park, Jinkook Kim, Junsang Park, Junho An, Kwanglo Lee, Wonsun Jeong, Sangil Chon, Kwonwoo Ha, Myungkyu Han, and Sungroh Yoon. Towards high generalization performance on electrocardiogram classification. 2021.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 1997.

[18] Jianyuan Hong, Hua-Jung Li, Chung chi Yang, Chih-Lu Han, and Jui chien Hsieh. A clinical study on atrial fibrillation, premature ventricular contraction, and premature atrial contraction screening based on an ecg deep learning model. 2022.

[19] Shenda Hong, Yuxi Zhou, Junyuan Shang, Cao Xiao, and Jimeng Sun. Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review, 2020.

[20] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. 1982.

[21] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[22] Rui Hu, Jie Chen, and Li Zhou. A transformer-based deep neural network for arrhythmia detection using continuous ecg signals. 2022.

[23] Melanie Humphreys. *Nursing the Cardiac Patient*. 2013.

[24] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning, 2018.

[25] Naimul Khan and Md Niaz Imtiaz. Pan-tompkins++: A robust approach to detect r-peaks in ecg signals, 2022.

[26] Lingxiao Meng, Wenjun Tan, Jiangang Ma, Ruofei Wang, Xiaoxia Yin, and Yanchun Zhang. Enhancing dynamic ecg heartbeat classification with lightweight transformer model. 2022.

[27] Petr Nejedly, Adam Ivora, Radovan Smisek, Ivo Viscor, Zuzana Koscova, Pavel Jurak, and Filip Plesinger. Classification of ecg using ensemble of residual cnns with attention mechanism. 2021.

[28] Petr Nejedly, Adam Ivora, Ivo Viscor, Zuzana Koscova, Radovan Smisek, Pavel Jurak, and Filip Plesinger. Classification of ecg using ensemble of residual cnns with or without attention mechanism. 2022.

[29] Jiapu Pan and Willis J. Tompkins. A real-time qrs detection algorithm. 1985.

[30] Adam G. Polak, Bartłomiej Klich, Stanisław Saganowski, Monika A. Prucnal, and Przemysław Kazienko. Processing photoplethysmograms recorded by smartwatches to improve the quality of derived pulse rate variability. 2022.

[31] Matthew A Reyna, Nadi Sadr, Erick A Perez Alday, Annie Gu, Amit J Shah, Chad Robichaux, Ali Bahrami Rad, Andoni Elola, Salman Seyedi, Sardar Ansari, Hamid Ghanbari, Qiao Li, Ashish Sharma, and Gari D Clifford. Will two do? varying dimensions in electrocardiography: The physionet/computing in cardiology challenge 2021. 2021.

[32] Estela Ribeiro, Felipe Dias, Quenaz Soares, Jose Krieger, and Marco Gutierrez. Deep learning approach for detection of atrial fibrillation and atrial flutter based on ecg images. 2023.

[33] Estela Ribeiro, Quenaz Bezerra Soares, Felipe Meneguitti Dias, Jose Eduardo Krieger, and Marco Antonio Gutierrez. Can deep learning models differentiate atrial fibrillation from atrial flutter? 2024.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[36] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. 2018.

[37] Apoorva Srivastava, Ajith Hari, Sawon Pratiher, Sazedul Alam, Nirmalya Ghosh, Nilanjan Banerjee, and Amit Patra. Channel self-attention deep learning framework for multi-cardiac abnormality diagnosis from varied-lead ecg signals. 2021.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2014.

[39] R.G. Tieleman, Y. Plantinga, D. Rinkes, G.L. Bartels, J.L. Posma, R. Cator, C. Hofman, and R.P. Houben. Validation and clinical use of a novel diagnostic device for screening of atrial fibrillation. 2014.

[40] Tomas Vicar, Jakub Hejc, Petra Novotna, Marina Ronzhina, and Oto Janoušek. Ecg abnormalities recognition using convolutional network with global skip connections and custom loss function. 2020.

[41] Tomas Vicar, Petra Novotna, Jakub Hejc, Oto Janousek, and Marina Ronzhina. Cardiac abnormalities recognition in ecg using a convolutional network with attention and input with an adaptable number of leads. 2021.

[42] Jibin Wang. Automated detection of atrial fibrillation and atrial flutter in ecg signals based on convolutional and improved elman neural network. 2019.

[43] Ziqiang Wang, Kun Wang, Xiaozhong Chen, Yefeng Zheng, and Xian Wu. A deep learning approach for inter-patient classification of premature ventricular contraction from electrocardiogram. 2024.

[44] Nima L Wickramasinghe and Mohamed Athif. Multi-label cardiac abnormality classification from electrocardiogram using deep convolutional neural networks. 2021.

[45] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[46] Genshen Yan, Shen Liang, Yanchun Zhang, and Fan Liu. Fusing transformer model with temporal features for ecg heartbeat classification. 2019.

[47] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022.

[48] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. 2017.

[49] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. 2017.

[50] Zibin Zhao. Transforming ecg diagnosis: An in-depth review of transformer-based deeplearning models in cardiovascular disease detection, 2023.

Appendix A

Implementation

```
# PyTorch implementation of the proposed approach (for single-lead ECG classification):
# 3 blocks: Multi-scale Convolutional Network
# 3 blocks: Transformer Encoder

# Multi-scale Convolutional Network part

class ConvolutionLayer(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size=5):
        super().__init__()
        self.conv = nn.Conv1d(in_channels, out_channels, kernel_size, padding='same')
        self.bn = nn.BatchNorm1d(out_channels)
        self.leaky_relu = nn.LeakyReLU()
        self.dropout = nn.Dropout(0.1)

    def forward(self, x):
        x = self.conv(x)
        x = self.bn(x)
        x = self.leaky_relu(x)
        x = self.dropout(x)
        return x

class MultiScaleKernelsBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.conv3 = ConvolutionLayer(in_channels, out_channels, kernel_size=3)
        self.conv5 = ConvolutionLayer(in_channels, out_channels, kernel_size=5)
        self.conv9 = ConvolutionLayer(in_channels, out_channels, kernel_size=9)
        self.conv15 = ConvolutionLayer(in_channels, out_channels, kernel_size=15)

    def forward(self, x):
        conv3 = self.conv3(x)
        conv5 = self.conv5(x)
        conv9 = self.conv9(x)
        conv15 = self.conv15(x)
```

```
        concatenated = torch.cat([conv3, conv5, conv9, conv15], dim=1)
        return concatenated

class ResidualCnnBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.residual_conv = ConvolutionLayer(in_channels, out_channels*4)
        self.multiscale_cnn_block = MultiScaleKernelsBlock(in_channels, out_channels)
        self.average_pooling = nn.AvgPool1d(kernel_size=2)

    def forward(self, x):
        input = self.residual_conv(x)
        x = self.multiscale_cnn_block(x)
        residual = x + input
        output = self.average_pooling(residual)
        return output

# Transformer Encoder part

class LayerNormalization(nn.Module):
    def __init__(self, parameters_shape, eps=1e-5):
        super().__init__()
        self.parameters_shape=parameters_shape
        self.eps=eps
        self.gamma = nn.Parameter(torch.ones(parameters_shape))
        self.beta =  nn.Parameter(torch.zeros(parameters_shape))
    def forward(self, inputs):
        dims = [-(i + 1) for i in range(len(self.parameters_shape))]
        mean = inputs.mean(dim=dims, keepdim=True)
        var = ((inputs - mean) ** 2).mean(dim=dims, keepdim=True)
        std = (var + self.eps).sqrt()
        y = (inputs - mean) / std
        return self.gamma * y  + self.beta

class FeedForward(nn.Module):
    def __init__(self, emb_dim, hidden, drop_prob=0.1):
        super().__init__()
        self.linear1 = nn.Linear(emb_dim, hidden)
        self.linear2 = nn.Linear(hidden, emb_dim)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=drop_prob)
    def forward(self, x):
        x = self.linear1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.linear2(x)
        return x
```

```
class MultiHeadAttention(nn.Module):
    def __init__(self, emb_dim, num_heads):
        super().__init__()
        assert emb_dim % num_heads == 0,
            "Embedding dimension must be divisible by number of heads."
        self.num_heads = num_heads
        self.head_emb_dim = emb_dim // num_heads

        # Note, that the query, key and value matrices
        # that project the embeddings are not shared.
        self.w_q = nn.ModuleList([nn.Linear(self.head_emb_dim, self.head_emb_dim)
                   for _ in range(num_heads)])
        self.w_k = nn.ModuleList([nn.Linear(self.head_emb_dim, self.head_emb_dim)
                   for _ in range(num_heads)])
        self.w_v = nn.ModuleList([nn.Linear(self.head_emb_dim, self.head_emb_dim)
                   for _ in range(num_heads)])
        self.w_o = nn.Linear(emb_dim, emb_dim)

    def _scaled_dot_product(self, q, k, v):
        d_k = q.size()[-1]
        attention_values = torch.matmul(q, k.transpose(-1, -2))
        scaled_attention_values = attention_values / sqrt(d_k)
        softmaxed_attention_values = softmax(scaled_attention_values, dim=-1)
        return torch.matmul(softmaxed_attention_values, v)

    def forward(self, x):
        batch_size, seq_len, emb_dim = x.size()
        q = torch.cat([layer(x[:, :, i*self.head_emb_dim:(i+1)*self.head_emb_dim])
            for i, layer in enumerate(self.w_q)], dim=-1)
        k = torch.cat([layer(x[:, :, i*self.head_emb_dim:(i+1)*self.head_emb_dim])
            for i, layer in enumerate(self.w_k)], dim=-1)
        v = torch.cat([layer(x[:, :, i*self.head_emb_dim:(i+1)*self.head_emb_dim])
            for i, layer in enumerate(self.w_v)], dim=-1)
        q = q.view(batch_size, seq_len, self.num_heads, self.head_emb_dim).transpose(1, 2)
        k = k.view(batch_size, seq_len, self.num_heads, self.head_emb_dim).transpose(1, 2)
        v = v.view(batch_size, seq_len, self.num_heads, self.head_emb_dim).transpose(1, 2)
        values = self._scaled_dot_product(q, k, v)
        values = values.transpose(1, 2).contiguous().view(batch_size, seq_len, emb_dim)
        return self.w_o(values) # additional output projection

class EncoderLayer(nn.Module):
    def __init__(self, emb_dim, ffn_hidden, num_heads, drop_prob):
        super().__init__()
        self.attention = MultiHeadAttention(emb_dim=emb_dim, num_heads=num_heads)
        self.dropout1 = nn.Dropout(p=drop_prob)
        self.norm1 = LayerNormalization(parameters_shape=[emb_dim])
        self.ffn = FeedForward(emb_dim=emb_dim, hidden=ffn_hidden, drop_prob=drop_prob)
        self.dropout2 = nn.Dropout(p=drop_prob)
        self.norm2 = LayerNormalization(parameters_shape=[emb_dim])
```

```
    def forward(self, x):
        residual_x = x
        x = self.attention(x)
        x = self.dropout1(x)
        x = self.norm1(x + residual_x)
        residual_x = x
        x = self.ffn(x)
        x = self.dropout2(x)
        x = self.norm2(x + residual_x)
        return x

class TransformerEncoder(nn.Module):
    def __init__(self, seq_len, num_clas, emb_dim, ffn_hidden, num_heads,
                 drop_prob, num_layers, pos_enc):
        super().__init__()
        self.positional_encoding = pos_enc
        self.layers = nn.Sequential(*[EncoderLayer(emb_dim, ffn_hidden, num_heads, drop_prob)
                      for _ in range(num_layers)])
        self.flatten = nn.Flatten()
        self.output_layer = nn.Linear(seq_len * emb_dim, num_clas)
        self.sigmoid = nn.Sigmoid()

    def _get_positional_encoding(self, seq_len, emb_dim):
        position = np.arange(seq_len)[:, np.newaxis]
        div_term = np.exp(np.arange(0, emb_dim, 2) * -(np.log(10000.0) / emb_dim))
        pe = np.zeros((seq_len, emb_dim))
        pe[:, 0::2] = np.sin(position * div_term)
        pe[:, 1::2] = np.cos(position * div_term)
        pe = pe[np.newaxis, :]
        return torch.tensor(pe, dtype=torch.float32)

    def forward(self, x):
        if self.positional_encoding:
            x = x + self._get_positional_encoding(x.shape[1], x.shape[2])
        x = self.layers(x)
        return x
```

```
# Combined model

class MultiScaleKernelsTransformerEncoder(nn.Module):
    def __init__(self, num_classes, ffn_hidden, num_heads, drop_prob,
                 num_encoder_blocks, positional_encoding=False):
        super().__init__()
        self.multiscalecnnblock1 = ResidualCnnBlock(1, 16)
        self.multiscalecnnblock2 = ResidualCnnBlock(64, 32)
        self.multiscalecnnblock3 = ResidualCnnBlock(128, 64)
        self.transformerencoder = TransformerEncoder(256, num_classes, 250,
            ffn_hidden, num_heads, drop_prob, num_encoder_blocks, positional_encoding)
        self.global_average_pooling = nn.AdaptiveAvgPool1d(1)
        self.output = nn.Linear(256, num_classes)

    def forward(self, x):
        # Input shape: (batch, 1, 2000) (based on below example initialization)
        x = self.multiscalecnnblock1(x)
        # shape: (batch, 64, 1000) -> 1000, because in each block average pooling is applied.
        # Note: Although, multiscalecnnblock1 is initialized with 16 output channels,
        # 4 different kernel sizes (3, 5, 9 and 15) are used, thus the concatenated
        # outputted channels sum to 64, likewise in the following two blocks, there 128 and 256.
        x = self.multiscalecnnblock2(x)
        # shape: (batch, 128, 500)
        x = self.multiscalecnnblock3(x)
        # shape: (batch, 256, 250)
        # Input for Transformer encoder blocks:
        # sequence_of_channels (256) x channel_dimension (250),
        # splitted into 10 heads (here 10 heads are used), with dimension 25 each.
        # Note, that the transposing step of the channels block
        # is not required here, as the correct shape is already present.
        x = self.transformerencoder(x) # (contains 3 encoder blocks)
        # shape: (batch, 256, 250)
        x = self.global_average_pooling(x).squeeze(-1)
        # shape: (batch, 256)
        x = torch.sigmoid(self.output(x))
        # Output shape: (batch, 26)
        return x

# Input data shape and model initialization example

input_data = torch.rand(32, 1, 2000) # 32 = batch size, 2000 = signal length
model = MultiScaleKernelsTransformerEncoder(num_classes=26, ffn_hidden=24, num_heads=10,
        drop_prob=0.1, num_encoder_blocks=3, positional_encoding=False)
```

```
# == Further code ==

# Adopted custom loss function (CL) - 3 classes example

x = torch.ones((3, 3), dtype=torch.float32)
challenge_weights = torch.tensor([[1.0, 0.5, 0.5],
                                  [0.5, 1.0, 0.5],
                                  [0.5, 0.5, 1.0]])
def CL(T, P): # Batch of true (binary) and predicted (continous) labels
    T = T.float() # e.g. [[1, 0, 0], [1, 0, 1]]
    P = P.float() # e.g. [[1., 0.3, 0.2], [0.4, 0.1, 0.8]]
    N = (T + P - T * P)
    N = torch.matmul(N, x) + 1e-6  # Small number added to avoid division by zero
    A = torch.matmul(T.transpose(1, 0), P / N)
    CL = challenge_weights * A
    # see: https://github.com/physionetchallenges/evaluation-2021/blob/main/weights.csv
    return torch.sum(CL)

# Modified multi-label confusion matrix
# Below code has nothing to do with above custom loss function, but was
# implemented and used for multi-label confusion matrix plots in the evaluation.

modified_confusion_matrix = np.zeros((num_classes, num_classes), dtype=int)
for true, pred in zip(y_true, pred_binary):
    for i in range(num_classes):
        for j in range(num_classes):
            if true[i] == 1:
                if i == j and pred[j] == 1:
                    confusion_matrix[i, j] += 1  # True Positive
                elif pred[j] == 1 and true[j] != 1:
                    confusion_matrix[i, j] += 1  # False Positive that only
                                                 # affect non true classes
```