Combining Machine Learning and Deep Learning Techniques for Multi-Label Classification of ECG with Variable Number of Leads

Tesi di Laurea Magistrale in
Ingegneria Biomedica - Tecnologie Elettroniche

Author: **Andrea Sansonetti**

Student ID: 953624
Advisor: Prof. Luca Mainardi
Co-advisors: Prof.ssa Valentina Corino, Guadalupe García-Isla
Academic Year: 2020-21

Abstract

The work presented in this thesis comprises the elaboration and evaluation of an ECG automatic classifier, which is able to assign multiple pathologies to recordings with a variable number of leads. The aim is to explore a Neural Network structure and its training procedure which are able to successfully exploit both Machine Learning and Deep Learning features in the same model. The data were selected from the multi-source dataset of the CinC/PhysioNet Challenge 2021, considering only 10 seconds, 500 Hz signals for a total of 84176 recordings. Up to 26 classes could be assigned to each track simultaneously. Some pathologies occurred more frequently than others, making the learning of the model biased towards them. To reduce this unbalancing, three subsets were created by dividing in three the Normal Sinus Rhythm and Sinus Bradycardia ECGs, which were the most predominant labels. Such subsets were exploited to train as many different models with the same architecture, used to create an ensemble which labels the samples by majority voting. The models comprised a "deep" branch, which extracted the signal's morphological features, and a "wide" branch, responsible for compressing and selecting the 20 handcrafted features. The output features of both branches were concatenated and fed to the final layer to produce 26 predictions, one for each class. A three step training was employed, inspecting its capability to prevent overfitting in the developed architecture: at first, only the deep part was trained, secondly the wide branch was instructed taking the deep features into account during the process, and finally a finetuning of the deep components was performed considering the wide parameters. The results obtained in the local test set, measured in terms of Challenge Metric (ranging from -1 to 1, worst to best) after the three step training, was 0.705 and 0.674 for 12 and 2-lead models respectively. This evaluation allows to assert that the integration of Deep Learning and handcrafted features is successful in improving the generalization capability, as well as in making the classification more explainable with respect to a pure Deep Learning process.

Keywords: ECG classification, Deep Learning, Machine Learning, variable leads, PhysioNet Challenge

Abstract in lingua italiana

Il lavoro presentato in questa tesi riguarda l'elaborazione e la valutazione di un classificatore automatico di ECG, capace di assegnare più patologie simultaneamente a tracciati con numero variabile di canali. Lo scopo è l'esplorazione della struttura di una Rete Neurale e di una procedura di addestramento della stessa tali da riuscire a integrare efficacemente elementi di Machine Learning e Deep Learning nello stesso modello. I dati usati sono stati selezionati tra i segnali provenienti dai diversi dataset considerati nella CinC/PhysioNet Challenge 2021, includendo soltanto le acquisizioni di 10 secondi campionate a 500 Hz, per un totale di 84176 tracciati. Ad ogni ECG sono state associate fino a 26 classi contemporaneamente. Alcune patologie erano presenti più frequentemente di altre, rendendo l'addestramento del modello sbilanciato verso di esse. Per ridurre tale sbilanciamento, sono stati realizzati tre sottoinsiemi di dati riducendo a un terzo il numero di segnali associati al Ritmo Normale Sinusale e alla Bradicardia Sinusale. Tali sottogruppi sono stati usati per allenare altrettanti modelli diversi con la stessa architettura, così da sfruttarli per fare un ensemble che assegna l'anomalia cardiaca per "majority voting". Il modello era formato da un ramo deep, che estrae le caratteristiche morfologiche dell'ECG, e da un ramo wide, responsabile della compressione e selezione delle 20 proprietà estratte a mano in precedenza. Gli attributi così identificati e restituiti dalle due parti sono concatenati e dati al layer finale per produrre 26 predizioni, una per ogni classe. Per allenare la rete è stata impiegata una procedura a tre fasi, indagandone la capacità di prevenire l'overfitting nell'architettura sviluppata: per prima cosa, è stato istruito solo il ramo "deep", quindi la parte "wide" è stata addestrata tenendo conto delle caratteristiche identificate nel ramo "deep", ed infine è stato eseguito un finetuning delle componenti "deep" considerando i parametri "wide". I risultati ottenuti sul test set locale, misurati in termini di Metrica della Challenge (che varia tra -1 e 1, dal peggiore al migliore), sono di 0.705 e 0.674 rispettivamente per i modelli a 12 e 2 canali. Questa valutazione permette di sostenere il vantaggio apportato dall'integrazione del Deep Learning con delle proprietà estratte a mano, poiché migliorano le capacità di generalizzazione del modello, e rendono la classificazione più comprensibile rispetto ad un processo di puro Deep Learning.

Parole chiave: Classificazione ECG, Deep Learning, Machine Learning, PhysioNet Challenge

Sommario

Secondo l'Organizzazione Mondiale della Sanità (OMS), le malattie cardiovascolari sono la prima causa di morte nel mondo [2]. Pertanto individuarle in tempo e agli stadi iniziali è fondamentale per la loro cura, e può evitare in alcuni casi la morte del paziente.

L'elettrocardiogramma (ECG) viene usato per identificare le anomalie elettriche del cuore [42]. La sua acquisizione segue delle procedure specifiche, e può variare a seconda del centro di acquisizione e dei dispositivi usati per eseguirla. Normalmente, un'acquisizione standard prevede la registrazione di tracciati a 12 canali per 30 secondi. Tuttavia, altri approcci, come l'uso dell'Holter [16], sono possibili, permettendo per esempio di acquisire finestre temporali più lunghe e con un numero variabile di canali [8, 48].

In tale contesto si colloca la PhysioNet/CinC Challenge 2021, che si pone l'obiettivo di studiare le performances dei classificatori automatici di ECG, in cui i segnali utilizzati possono anche essere acquisiti con un numero di canali ridotto [40]. Tale scopo si propone come evoluzione della Challenge dell'anno precedente, in cui veniva richiesto lo sviluppo di algoritmi per la classificazione automatica di tracciati ECG a 12 canali [39]. La complessità affrontata riguardava l'uso di un dataset eterogeneo, che comprendeva più di 80000 dati con più di 100000 casi, ottenuto raccogliendo campioni da 7 fonti differenti. Il modello richiesto dovrebbe perciò produrre classificazioni che non sono influenzate dall'origine del segnale considerato. La ricerca in questo campo è molto importante, poichè il contributo apportato potrebbe portare ad algoritmi con buone capacità diagnostiche, in grado di supportare i cardiologi nell'identificazione delle malattie cardiovascolari. In particolare, tali modelli potrebbero aiutare nello screening di prevenzione, andando a ridurre le tempistiche con cui si riceve una diagnosi, spesso molto lunghe per via della carenza di medici specialisti.

Stato dell'arte

I classificatori automatici possono essere divisi in due categorie: quelli basati sul Machine Learning (ML) e quelli basati sul Deep Learning (DL) [44].
La differenza principale è che i primi imparano a classificare i dati in ingresso partendo

da caratteristiche fornite al modello direttamente dal programmatore esterno, dopo una procedura di estrazione, elaborazione e selezione svolta a priori. Perciò l'addestramento di tali algoritmi fa si che essi imparino a riconoscere le classi da assegnare all'input partendo esclusivamente da queste proprietà, andando ad ottimizzare una funzione di costo.
Dall'altro lato, il Deep Learning sfrutta delle strutture matematiche in grado di estrarre le proprietà dell'input in maniera autonoma: perciò anche l'elaborazione, la selezione e l'utilizzo di tali caratteristiche è lasciato all'addestramento dell'algoritmo stesso. Questo può ridurre l'interpretabilità delle caratteristiche e del motivo per cui si raggiunge una determinata classificazione, ma spesso permette il raggiungimento di risultati più elevati.

Nell'ambito della classificazione degli ECG, il Deep Learning sta iniziando a prendere piede: numerosi studi che riguardano la classificazione di tale segnale mostrano accuratezze molto elevate. Ciononostante, i casi studio sono sempre stati su campioni ristretti, su un numero ridotto di patologie e sempre a classificazione singola, senza mai raggiungere lo stesso grado di complessità richiesto dalla PhysioNet Challenge [12]. Il lavoro proposto in questa tesi parte dal lavoro svolto dal team *PhysioNauts* nella Challenge, e lo porta ulteriormente avanti per migliorare le performances del classificatore di Deep Learning sviluppato.

L'idea principale attorno a cui ruota l'algoritmo prodotto parte dalla considerazione secondo cui le manifestazioni dell'attività elettrica del cuore possiedono una doppia natura implicita: alcune provocano un'alterazione *morfologica* del segnale ECG, altre influiscono sulle componenti *ritmiche* del battito cardiaco; spesso, entrambe le parti sono modificate. Prendendo spunto dai lavori precedenti di altri gruppi che hanno preso parte alla Challenge 2020, e sfruttando le due alterazioni causate nel segnale ECG dalle patologie cardiache, la Rete Neurale proposta è stata realizzata in due rami: il primo, in grado di focalizzarsi sulla morfologia dei tracciati cardiaci, sviluppato partendo da una ResNet con Attention Mechanism di Zhao et al. [50] e modificato per migliorarne le prestazioni. Il secondo ramo combina invece Machine Learning e Deep Learning, andando ad utilizzare grandezze ritmiche estratte dal segnale, e dandole in input ad una rete molto semplice che seleziona e combina in maniera automatica tali caratteristiche.

Materiali e Metodi

Tutti i partecipanti alla PhysioNet/CinC Challenge 2021 avevano a disposizione 88253 ECG, ciascuno associato ad una o più patologie prese da un bacino di 133 complessive. Di queste, soltanto 30 erano valutate nella Challenge; inoltre tale numero si riduce a 26 se si considera che 4 coppie di tali patologie erano considerate equivalenti. I dati considerati provenivano da 7 diversi centri di acquisizione, ognuno con le proprie caratteristiche e configurazioni hardware che hanno influito sulle proprietà del segnale finale. Per questo motivo sono state svolte una procedura di selezione ed elaborazione dei tracciati cardiaci prima di poterle dare in ingresso al modello: sono stati mantenuti solamente i segnali di 10 secondi e campionati a 500 Hz, per garantire degli input omogenei per l'algoritmo. Inoltre, le due patologie più frequenti, il Ritmo Normale Sinusale (SNR) e la Bradicardia Sinusale (SB), sono state divise per creare 3 sottogruppi contenenti ciascuno un differente terzo dei dati classificati con queste anomalie, e mantenendo tutti gli altri tracciati.

A seguire, è stata eseguita l'elaborazione dei dati, diversificata a seconda del blocco del modello considerato. Per la parte di riconoscimento morfologico, detta anche "**deep**", ovvero "profonda" in lingua italiana per richiamare la complessità di questa parte della rete e il numero di livelli, o "layers", utilizzati, l'elaborazione era volta a standardizzare e rimuovere le differenze presenti tra i segnali con origini differenti. Prima di tutto è stato eseguito il resampling a 500Hz di tutti i segnali, poi filtrati con un passabanda tra 1 e 47 Hz, quindi normalizzati con lo z-score per rimuovere il bias legato a diverse medie e deviazioni standard. Infine, se la lunghezza del segnale considerato era inferiore a 5000 campioni, veniva realizzata la tecnica dello zero-padding, altrimenti veniva presa una finestra casuale di 10 secondi.
Considerando invece la parte di identificazione delle caratteristiche ritmiche, detta blocco "**wide**", cioè "ampio" (contrapposto al ramo precedente per la maggiore semplicità della rete neurale che lo compone), l'elaborazione era volta all'estrazione delle grandezze numeriche temporali da utilizzare nella classificazione. Prima di tutto, è stato estratto il canale "II" e filtrato con un passabanda nel range $[3-45]$ Hz. Dopodichè sono stati identificati i picchi R per poter quindi calcolare gli intervalli RR e i grafici di Poincaré relativi a ciascun ECG. Infine, sono state elaborate le 20 grandezze fornite come input a questa parte del modello.

Guardando nel dettaglio l'architettura della Rete Neurale, il blocco "deep" è fatto da una Residual Network con Attention Mechanism dato da dei blocchi "Squeeze-and-Excitation" (blocchi SE). Tale tipo di rete si concentra sull'utilizzo dei cosiddetti "Blocchi Residuali" [19], grazie ai quali è possibile realizzare reti con molti livelli, necessari per raggiungere complessità elevate nelle caratteristiche elaborate dal modello. In particolare, le proprietà estratte si concentrano sulla morfologia degli input ricevuti. L'uso dei blocchi SE ha permesso di identificare proprietà morfologiche non solo nel singolo canale dell'ECG, ma anche tra i vari canali, per sfruttarne le interazioni nella classificazione. Inoltre, nei blocchi residuali sono stati aggiunti meccanismi di convoluzione con dilatazione, che hanno reso possibile individuare le proprietà morfologiche dei dati a diverse scale temporali.

Dall'altro lato, il blocco "wide" si sviluppa come una semplice Rete Neurale a imbuto, in cui ci sono 3 layers con 20, 15 e 10 neuroni, la cui funzione è di selezione e sintesi delle grandezze numeriche estratte e ricevute come input.

Gli output dei due rami vengono quindi concatenati e dati in pasto all'ultimo livello formato da 26 neuroni, che svolge la funzione di classificatore finale, restituendo le probabilità di appartenenza a ciascuna classe considerate dalla Challenge.

Un altro aspetto a cui si è data importanza è stato lo sviluppo di una procedura di addestramento della rete tale da ridurre l' "overfitting" legato alla differenza nel numero dei parametri della rete. Il termine "overfitting" identifica il fenomeno che si presenta quando i modelli imparano a riconoscere perfettamente i dati usati nell'allenamento, ma hanno poi prestazioni scarse nei confronti di dati diversi. Spesso questo coincide con il fatto che le caratteristiche individuate riguardo i dati usati nell'addestramento non sono in grado di generalizzare il problema, ma sono solo specifiche di quel particolare gruppo di campioni. Un'altra possibile causa può essere la non corretta ottimizzazione dei parametri nell'architettura. Data la differente complessità dei rami "deep" e "wide" utilizzati, è possibile che l'overfitting avvenga in tempi diversi tra le due parti, in quanto è diverso il numero di parametri da ottimizzare. Perciò è stata realizzata una procedura di addestramento in 3 fasi: le prime due fasi si sono concentrate sul far apprendere a ciascun blocco le proprietà che gli competevano senza essere disturbato dall'altro. La terza fase era invece focalizzata sulla sincronizzazione tra i due blocchi, apportando piccole modifiche ai pesi del blocco "deep" e tenendo in considerazione la parte "wide" addestrata.

Infine, dopo l'addestramento, è stato affrontato il problema dello sbilanciamento dei dati, considerando il numero di tracciati rispetto alle patologie presenti: essendoci disturbi che accadono più raramente di altri, la rete li predirrà molto meno di frequente e con probabilità molto basse, rendendo difficile la loro identificazione. Per compensare questo effetto è stata ideata una procedura di ottimizzazione delle soglie dei neuroni dell'ultimo layer, responsabile della produzione delle probabilità di classificazione. Tali soglie giocano un ruolo fondamentale nella classificazione, poichè sono confrontate con le probabilità finali per decidere se la classe è presente o meno. Esse sono state ottimizzate rispetto al valore di metrica della Challenge ottenuto: tale metrica è un'accuratezza generalizzata che penalizza i falsi positivi, perciò massimizzare questo valore dovrebbe portare ad un miglioramento delle performances evitando di produrre classificazioni errate. Tale ottimizzazione è stata effettuata sul dataset di validazione, distinto da quello di training, per garantire una generalizzazione maggiore nelle predizioni.

L'architettura proposta è stata quindi allenata con la procedura precedentemente indicata sui 3 sottoinsiemi identificati nella selezione dei dati, producendo 3 modelli con la stessa struttura. Essi sono stati poi utilizzati per svolgere una classificazione dei dati di test nella modalità "ensemble", o "complessiva" in italiano, per voto di maggioranza: i tre modelli producono ciascuno le proprie probabilità e classificano ogni dato, ma solo se almeno due algoritmi su tre stabiliscono la presenza di una patologia questa viene effettivamente assegnata a quell'ECG. Questa procedura viene usata per aumentare le capacità di generalizzazione del modello.

Risultati

La metrica della Challenge è un'accuratezza pesata, in cui anche le predizioni errate vengono premiate con un punteggio parziale a seconda del tipo di errore effettuato, a causa della somiglianza dell'effetto di alcune patologie sul tracciato cardiaco. Tuttavia, un termine di normalizzazione penalizza falsi positivi, perchè con l'aumentare del numero di malattie assegnate tale valore viene incrementato, riducendo così il punteggio finale.

I risultati di testing del modello finale in termini di metrica della Challenge dopo la terza fase di addestramento per i 12 e i 2 canali sono rispettivamente 0.705 e 0.674. Le prestazioni sono discrete in termini di valore assoluto dell'accuratezza, ma se confrontate con quelle ottenute dai modelli sviluppati durante le altre 2 fasi della Challenge, mostrano le potenzialità dei miglioramenti apportati. Durante la prima fase "non ufficiale", sono stati ottenuti punteggi di 0.520 e 0.505 sul test set valutato con un'architettura che includeva solo il ramo "deep" per i 12 e i 2 canali; quindi, nella seconda fase "ufficiale", grazie all'introduzione della parte "wide" ma senza la procedura di addestramento a tre fasi nè le convoluzioni con dilatazione sono stati ottenuti punteggi rispettivamente di 0.643 e 0.633 per i 12 e i 2 canali. Inoltre, nello stato dell'arte rappresentato dai lavori di altri gruppi che hanno preso parte alla PhysioNet Challenge non si riscontrano valori di accuratezza che si discostano significativamente da quelli raggiunti. Si invita il lettore a leggere la sezione "Results" dell'elaborato per poter comprendere a pieno le considerazioni e i paragoni fatti sui risultati prodotti.

Discussione e Conclusioni

Nello sviluppare il modello, diversi aspetti sono stati presi in considerazione e portati avanti.

Prima di tutto, la realizzazione di un modello in cui Machine Learning e Deep Learning fossero integrati aveva un duplice obiettivo: l'incremento delle capacità di riconoscimento

delle patologie che non alterano la morfologia degli ECG in maniera evidente, e la maggiore interpretabilità della classificazione ad un utente esterno tramite l'uso di proprietà calcolate in maniera esplicita.
Inoltre, secondo le richieste della Challenge è stata esplorata l'efficacia della classificazione degli input con numero di canali ridotto.
Un'altra novità messa a punto nella stesura dell'algoritmo è stato l'addestramento a fasi della rete, volto a un incremento delle prestazioni dei singoli blocchi del modello.
Infine, per trattare i problemi di sbilanciamento del dataset, diverse strategie sono state utilizzate prendendo spunto dai lavori precedenti sulla Challenge, ottimizzando le soglie dei neuroni nel layer finale e dividendo i gruppi di dati più grandi per realizzare un ensemble di modelli, che è generalmente più performante sul test set rispetto al classificatore singolo.

Tutti questi aspetti hanno contribuito positivamente, portando il modello a raggiungere prestazioni paragonabili a quelle presenti nello stato dell'arte, ed hanno aperto a nuovi possibili approcci nella classificazione degli ECG tramite l'uso di classificatori parzialmente espliciti. Inoltre, dai risultati ottenuti si può osservare come il calo di prestazioni nel caso a canali ridotti non sia eccessivo. Quindi, apportando ulteriori migliorie e aumentando la capacità diagnostica del modello, è possibile che si raggiungano livelli tali da poter impiegare il classificatore anche in ambiti in cui non è possibile registrare un ECG standard a 12 canali.

I lavori futuri dovrebbero pertanto concentrarsi nell'identificare nuove idee al fine di migliorare le prestazioni, andando per esempio ad aumentare le caratteristiche usate dal ramo "wide", o ancora modificandone la struttura per renderla più efficace nella selezione ed integrazione con il blocco "deep".

Executive Summary

According to the World Health Organization (WHO), cardiovascular diseases are the world's first cause of death [2]. Thus, their early detection is essential for a correct treatment, allowing in some cases to avoid the patient's death.

Electrocardiogram (ECG) is exploited to identify the electrical abnormalities of the heart [42]. Its acquisition follows specific procedures, and varies according to the acquisition center and the hardware configuration used. A standard acquisition is performed by acquiring the electrical signal with 12 channels for 30 seconds. However, other approaches, such as the use of an Holter [16], are possible, allowing for instance to acquire longer temporal windows with a variable number of leads [8, 48].

In such context, the PhysioNet/CinC Challenge 2021 takes place, with the purpose of studying the performances of ECG automatic classifiers, in which the used signals may also present a reduced number of leads [40]. Such objective is the prosecution of the previous year's Challenge, where the development of 12 leads ECG classifiers was required [39]. The complexity faced regarded the heterogeneity of the dataset used, which comprised more than 80000 samples presenting more than 100000 cases, collected from 7 different sources. The model required should be able to produce classifications which are not influenced by the origin of the recording taken into account. Research in this field is very important, because the new discoveries could bring efficient algorithms, such that they could support cardiologists in the identification of cardiovascular diseases. In particular, such models could help in screening for prevention by reducing the diagnostic time, often very long due to limited availability of expert clinicians.

State of the Art

Automatic classifiers can be divided in two categories: those based on Machine Learning (ML) and the ones relying on Deep Learning (DL) [44].
The main difference between the two is that the former learns to classify the inputs starting from features directly given by the programmer to the learner, after an a-priori procedure of extraction, elaboration and selection. Thus, such algorithms are trained so

that they recognize the labels of classification exclusively using such features, through the optimization of a loss function.
On the other hand, Deep Learning exploits mathematical structures which are able to autonomously extract the input characteristics: so the elaboration, selection and use of these features is left to be learnt by the algorithm itself. This aspect may reduce the interpretability of such features, as well as the explainability of the final classification, but it also allows to reach better results.

In ECG classification, Deep Learning is taking place thanks to the different works proposing classifiers with high accuracy. However, such studies have always been developed on a small sample size, with a reduced number of pathologies and with single classification, never reaching the same degree of complexity as the one proposed in the PhysioNet Challenge [12]. The research proposed in this thesis starts from the work developed by the team *PhysioNauts* during the Challenge, pushing it forward to improve the developed Deep Learning classifier's performances.

The main idea of the proposed algorithm is that the manifestation of heart electrical activities have an implicit double nature: some of them induce a *morphological* alteration of the ECG signal, while others provoke a *rhythmic* change in the heartbeat; often, both are influenced.
Starting from other groups' works in the previous 2020 Challenge, and exploiting this double alteration of ECG signal provoked by the pathologies, the provided solution was a Neural Network composed by two branches: the first one, which focuses on the morphology of cardiac recordings, was developed starting from a ResNet with Attention Mechanism of Zhao et al. [50], and further improved. The second branch combines together Machine Learning and Deep Learning, by employing rhythmic features extracted from the signal, which are fed to a simple network that automatically selects and combines them.

Materials and Methods

All the participants in the PhysioNet/CinC Challenge 2021 had at their disposal 88253 ECGs, each one associated with one or more pathologies taken from a total of 133. Out of those, only 30 were evaluated in the Challenge; moreover, this number can be reduced to 26 considering the fact that 4 couples of such pathologies were considered equivalent. Such data were originated from 7 different centers of acquisition, each one with its own characteristics and hardware configurations which influenced over the signal properties. For this reason a selection and elaboration of ECG recordings was performed before they were fed to the model: only 10 seconds 500 Hz signals were kept, to guarantee homogeneous inputs for the algorithm. Moreover, the two most frequent pathologies, the Normal Sinus Rhythm (NSR) and the Sinus Bradycardia (SB), were divided to create 3 subsets, each containing a different third of data labelled with such anomalies, and keeping all the other recordings.

Data processing followed, distinguishing the procedure according to the considered block of the model. For the morphological branch, also named "**deep**" to recall the complexity of this part of the network, as well as the number of layers employed, the processing was aimed to the standardization and difference removal among the data with different origin. First of all, all signals were resampled at 500 Hz, then filtered with a passband filter between 1 and 47 Hz, and normalized with z-score to remove the mean and standard deviation biases. Finally, if the considered signal was shorter than 5000 samples, zero-padding was applied, else a random 10 seconds window was taken.
Considering the part of rhythmic identification, called "**wide**" branch - juxtaposed with the previous one for the simplicity of the employed Neural Network - the processing of data was focused on the extraction of the numerical features to be used for the classification. First of all, the lead "II" was selected and then filtered in range $[3-45]$ Hz with a passband filter. Then, the R peaks were identified to compute the RR intervals and Poincarè images of each ECG. Finally, the 20 temporal characteristics were elaborated and fed to this part of the model.

Watching in details the Neural Network's architecture, the "deep" part is composed by a Residual Network with Attention Mechanism implemented through "Squeeze-and-Excitation" (SE) blocks. Such Network focuses on the use of the so called Residual Blocks [19], which make it possible to realize structures with numerous layers, necessary to reach high complexity in the features computed by the model. In particular, the extracted properties focus on the input morphology. The use of the SE blocks allowed the identification of morphological attributes not only in the single lead, but also among the channels of an ECG, to exploit their interactions for the classification. Moreover, dilation mechanisms in the convolutions performed in the Residual blocks were added, making it possible to recognize the morphological features at different time scales.

On the other hand, the "wide" block is developed as a simple bottleneck Neural Network, with its 3 layers made of 20, 15 and 10 neurons, whose function is to select and summarize the numerical handcrafted features received as input.

The two branches' outputs were then concatenated and fed to the last layer, made up of 26 neurons, which act as final classifier, returning the probabilities of each class considered by the Challenge.

Another important aspect considered was the implementation of the network's training procedure such that it would reduce the "overfitting" due to the difference in the dimension of the model's parameters. The term "overfitting" identifies the phenomenon of getting perfect recognition of training data classes, but bad performances on a different datasets. This often happens when the identified features of data used during training are not able to generalize the problem, but only mimik the specific group of samples. Another possible issue could be the not correct optimization of the model's parameters. Given the different complexity of the "deep" and "wide" branches exploited, it is possible that overfitting would take place with different timing in the two parts, because the number of weights to be optimized changes. Thus a 3 phases procedure was realized: the first two were focused on making each block learn its parameters without being hintered by the other one. The third phase was focused on the synchronization of the two blocks, making slight changes on the "deep" part while taking into account the trained "wide" branch.

At last, after the training, the data unbalancing problem was faced, considering the number of recordings with respect to the pathologies: some diseases happened less frequently with respect to others, thus the network would predict them rarely and with very low probabilities, making them difficult to be identified. To compensate this effect, an optimization procedure over the thresholds of the neurons belonging to the final classifier layer was implemented. These thresholds played a fundamental role in classification, because they are compared with the final probabilities to decide whether a class is present or not. They were optimized according to the Challenge Metric obtained: such metric is a generalized accuracy which penalized false positives, thus maximizing this value would bring an improvement of performances without producing misclassifications. Such optimization was performed, as part of hyperparameter tuning, over the validation dataset, distinguished from the training one to guarantee a better generalization in predictions.

The proposed architecture was trained with the previously illustrated procedure over the 3 subsets of data identified during the data selection, producing 3 models with the same structure. They were used for the classification of test data with an "ensemble" modality by majority voting: the three models returned their own probability and classified each ECG, but only if at least 2 out of the 3 algorithms gave the same pathology, the label was effectively assigned. This last procedure was used to increase the model generalization.

Results

The Challenge Metric is a weighted accuracy, which rewards misclassification with a partial score according to the error, due to the similiarity of the effect of some pathologies on the ECG. However, a normalization term discourages the false positives, because it increases with the number of diseases assigned, thus reducing the final score.

The testing phase results scored with Challenge Metric after the third step of training for the 12 and 2-leads models were respectively 0.705 and 0.674. The performances are discrete in terms of absolute value of accuracy. However, if compared with the results obtained over the models of the other 2 phases of the Challenge, they show the potentiality of the introduced improvements. In the first "unofficial" phase, scores of 0.520 and 0.505 were obtained on the test set for the 12 and 2-leads evaluated with an architecture with only the Deep branch; then, in the second "official" phase, with the introduction of the wide branch without the three step training nor the dilated convolutions a scoring of 0.643 and 0.633 was achieved. Moreover, by observing the other models which took part to the PhysioNet Challenge and comparing them with the acquired scores, no particular discrepancies can be observed. The reader is invited to read the section "Results" of the elaborate to fully appreciate the considerations done over the obtained results.

Discussions e Conclusions

By developing the model, different aspects were taken into account and brought forth.

First of all, the integration of Machine Learning and Deep Learning in a single model had a double aim: the increase in the recognition capability of pathologies which do not explicitly alter only the morphology of an ECG, as well as the better interpretability of the classification to the external user, thanks to the use of explicit handcrafted features. Moreover, following the Challenge questions, the effectiveness of classification of inputs with reduced leads was explored.
Another novelty introduced in the implementation of the algorithm was the training in steps, which wanted to improve the performances of the single branches of the model.
Finally, to deal with the dataset unbalancing problems, different strategies were exploited by taking inspiration from previous Challenge works, as the threshold optimization and the division of the largest group of data for the realization of an ensemble of models, which usually performs better over the test set with respect to a single classifier.

All these aspects positively contributed to the improvement of model's performances, bringing them to a level which is comparable to the state of the art, and opening to new possible approaches in the ECG classification with the use of partially explicit classifiers. Moreover, from the obtained results, it can be seen how the drop in performances in the reduced lead set is not excessive. So, by further improving the model's diagnostic capability, it is possible to reach a point in which a classifier can be used also when a standard 12-leads ECG is not available.

Future works should focus on the identification of new ideas to improve performances, for instance by increasing the number of features used in the "wide" branch, or also changing its structure to make the selection more effective and better integrate it with the "deep" block.

Contents

Abstract **i**

Abstract in lingua italiana **ii**

Sommario **iv**

Executive Summary **x**

Contents **xvi**

Introduction **1**

1 State of the Art **4**

1.1 Artificial Neural Networks . . . 5

1.1.1 Perceptron . . . 5

1.1.2 Multi-Layer Perceptron . . . 7

1.1.3 Feed-forward Neural Network . . . 7

1.1.4 Feedback Neural Networks . . . 9

1.1.5 Recurrent Neural Networks . . . 10

1.1.6 Convolutional Neural Networks . . . 10

1.1.7 Attention Mechanism . . . 14

1.2 Network training . . . 15

1.2.1 Learning rules . . . 16

1.2.2 Overfitting . . . 17

1.2.3 Ensemble of models . . . 20

1.3 Neural Networks for ECG classification . . . 21

2 Materials and Methods **24**

2.1 Data description . . . 24

2.1.1 Electrocardiogram . . . 24

2.1.2 PhysioNet/CinC Challenge Dataset 27
2.1.3 Cardiac abnormalities in PhysioNet Challenge 31
2.2 Data processing 33
2.3 Network Architecture 35
2.3.1 Deep branch 36
2.3.2 Wide branch 38
2.4 Training procedure 39
2.5 Ensemble evaluation 41

3 Results **44**
3.1 Challenge Metric 44
3.2 PhysioNet Challenge evaluation 46
3.2.1 Unofficial Phase Results 47
3.2.2 Official Phase Results 48
3.2.3 Focus Issue Phase Results 49

4 Conclusions and future developments **59**

Bibliography **62**

A Appendix A - Model code **68**

List of Figures **87**

List of Tables **88**

Introduction

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the first cause of death in the world [2]. Early detection of these pathologies plays a key role in their treatment, and can avoid fatal situations.

Electrocardiogram (ECG) is used to identify the electrical abnormalities of the heart [42]. Its acquisition follows a specific procedure and depends on the centre of acquisition, as well as on the device used. Standard clinical ECG consists in 12-leads 30s recording, but other approaches, such as Holter [16] monitoring, may collect longer time windows with variable number of leads [8, 48].

In this context fits the PhysioNet/CinC Challenge 2021, whose aim is to explore the performances of automatic ECG classifiers on reduced sets of leads [40], prosecuting the previous year Challenge focal points [39]. Over 88000 ECG recordings were available for the model training, which needs to perform a multi-label classification of 30 different classes. The research in this field could provide good algorithms which may support effectively the clinicians and cardiologists in the diagnosis of CVDs, improving the chances of early detection and screening of such pathologies, and reducing the time required for such analysis.

Automatic classifiers can be divided in two main categories: Machine Learning (ML) classifiers and Deep Learning (DL) ones [44]. The main difference is that the first ones learn from handcrafted features that are elaborated, selected and fed to the algorithm by the programmer. The training will then make the model recognize the characteristics of each class using only these elements, through the optimization of a loss function. On the other hand, Deep Leaning exploits a network that extracts features on its own. This reduces the interpretability of the features, but also allows the classifier to understand by itself which features are the most important ones to optimize the parameters of classification.

The Deep Learning approach is starting to take place in the field of ECG classification, with many successful studies in the classification of different pathologies, but never at the same level of complexity posed by the Challenge [12]: usually, the research focuses on a smaller number of pathologies, and the predictions do not give back multiple labels.

The work presented in this thesis starts from the contribution of the *PhysioNauts* team to the PhysioNet Challenge, and tries to further expand the performances of the Deep Learning classifier developed.

The concept behind the network architecture is based on the consideration that the CVDs have an implicit double nature: some generate a *morphological* alteration of the signal, others affect the *rhythmic* component of the heartbeat; often, both are influenced. This idea was exploited by reconstructing a Neural Network made up of two parts: a first one that would recognize the shape of pathologies exploiting the channel relationships, and another that is focused on rhythmic characterization of the signal through handcrafted features. The first part is made of a modified Residual Network with Squeeze and Excitation blocks, as presented in the work by Zhao et al. [50], to which additional changes such as Dilation convolutions have been performed. This branch should effectively capture the morphological characteristics in the signal as well as among channels.

The second branch is made of a simple Fully Connected Neural Network made of three layers which takes as input the handcrafted features and select them. This approach exploits the ML feature extraction and DL automatic selection thanks to the Dense layers.

Finally, the outputs of the two parts is concatenated and fed to the classifier layer, which produces 26 probabilities: these probabilities are compared to thresholds computed for each label, returning the final multiple predictions.

The achieved results show that automatic classifiers can be effective even in complex problems such as the classification of 26 different pathologies, even though they are not very accurate. These performances are limited for different factors, such as the quality and numerosity of data. In fact, the number of samples of a specific class available in the training phase influences the ability to recognize it, and for some classes there were not enough data. Moreover, by understanding the nature of specific pathologies and implementing models which correctly entangles them, they can be better identified improving the overall performances. Also, to obtain good predictions, the model should be trained on a dataset where the labeling criterion is homogeneous on all the signals; in this case this specific requirement did not occur due to the nature of the Challenge to exploit all the datasets acquired from different centers. Relabelling the signals could have been useful, as shown by other groups in their work, but is a very time consuming methodology that requires expert cardiologists, often not available.

The chapters presented in this work are organized as follows:

- Chapter 1 is the analysis of the state of the art, comprising the previous studies that tried to approach the problem of automatic multi-label classification of ECGs. Here is also reported the basis from which this work started.

- Chapter 2 includes the Materials and Methods that were developed to produce the final classifier.
- Chapter 3 shows the final results, explaining them in details and making hypothesis on their meaning.
- Chapter 4 comprises the final Conclusions with pros and cons of the considered model, and expresses some of the future developments that could be followed starting from this work.

1 | State of the Art

Electrocardiogram (ECG) analysis is one of the most important tools for the diagnosis of cardiovascular diseases (CVDs), and can be of utmost importance in the early detection of the pathologies and their treatment, improving the outcome of surgery intervention and avoiding fatal events [17].

CVD diagnostic is mainly performed by expert physicians who need to take into account the patient's status, his medical history as well as the actual ECG signal recording to understand the overall condition of the subject. This procedure is highly time consuming, depends on the doctor's experience and can be susceptible of error if the person is not a specialist. Moreover, the need of expert physicians makes them overloaded with work, making it difficult to receive their opinion in time for the early diagnostic [12].

For this reason, automatic ECG classifiers have been developed to support this need of complex diagnostics and try to automatize the procedure, so that only in the cases that really need an in-depth analysis the experts are involved, reducing their load and optimizing the available resources of the hospital.

The history of classifiers begun with Machine Learning methods, that needed handcrafed features to analyze and classify the recordings. From this starting point Deep Learning algorithms started to take place in this field, thanks to their automatic and implicit feature extraction that reduced computational costs and improved the results of classification. Nowadays these are the most diffused classifiers, that have very good performances in the simplest cases, reaching an accuracy close to the one of an expert cardiologist [12].

The new frontier is the development of networks that are capable of successfully classify different and more complex pathologies simultaneously [40]. Multilabel classifiers could be a further step in the recognition of pathologies that interact with each other modifying the behavior of the heart.

1.1. Artificial Neural Networks

Deep Learning is a branch of Machine Learning based on Artificial Neural Networks (ANNs), mathematical structures inspired by the physiological studies of brain, which try to mimic its cells' behavior [6, 25].

1.1.1. Perceptron

The basic unit of an ANN is the **Perceptron**: this element receives one or multiple inputs which are weighted and integrated. This weighted sum is fed to an Activation Function, which is a mathematical function that transforms these inputs, returning the unit output [41, 45]. The Activation Function can include a Bias, that is a fixed input which modify the outcome of the neuron (Figure 1.1).

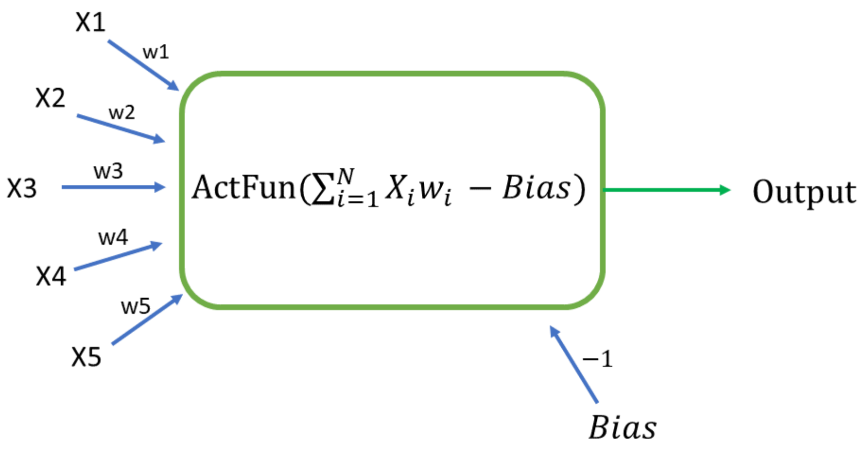
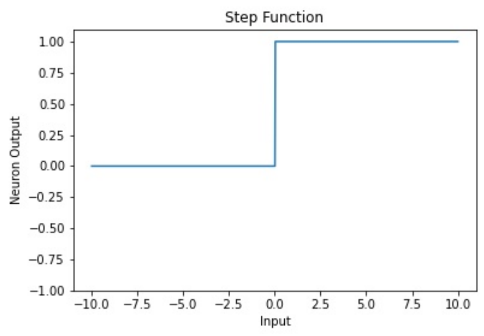
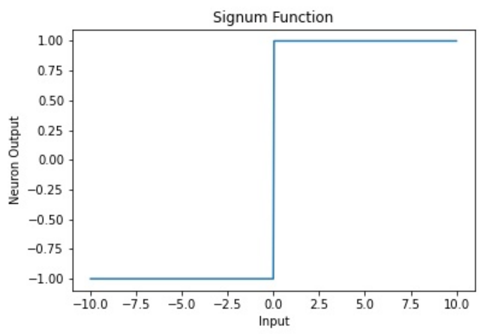


Figure 1.1: A basic Perceptron with 5 inputs X_i: each input is weighted and summed, then a bias is removed from this integration, and the result is given to the activation function. The result of these operations is the output of this unit.

The simplest Activation Function is the Step function, which gives a binary output: if the value fed to the function is smaller than 0.5 the output will be 0; else, it will be 1. As the function is not defined in 0.5, a convention is to be decided to understand which output to return if the value is exactly 0.5 (Figure 1.2a). If the range is between -1 and 1, the Signum activation function can be used instead (Figure 1.2b).



(a) Step Activation Function.

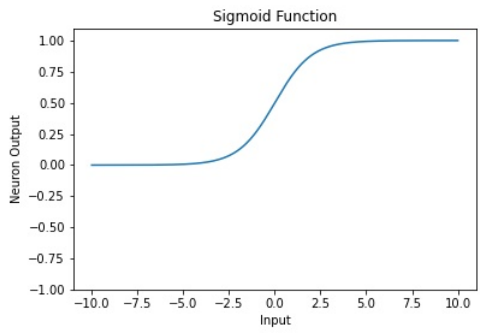


(b) Signum Activation Function.

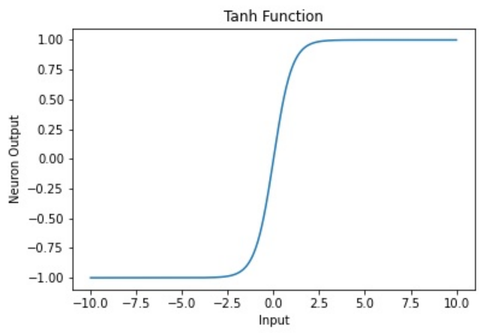
Figure 1.2: Binary Activation Functions of a simple Perceptron.

Starting from the basic binary functions (Figure 1.2), activation functions can be changed to have continuous nonlinear outputs in the range $[0; 1]$ or $[-1; 1]$ (Figure 1.3), reproduce complex behavior of the output in any range (Figure 1.4b) or simply return the input as it is after the integration (Figure 1.4a).

In particular, in Figure 1.3a is shown the Sigmoid activation function, which returns a continuous output in the range $[0; 1]$ with different slopes: around 0.5 the output follows almost linearly the input, but as it goes toward the extreme values it will reduce its speed, until it reaches saturation. The Hyperbolic Tangent activation function (Figure 1.3b) follows the same behavior, but its range is in $[-1; 1]$, so it includes negative values.



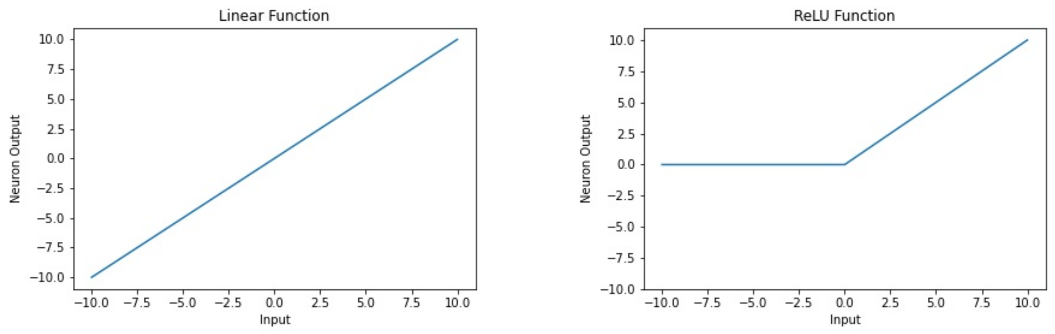
(a) Sigmoid Activation Function.



(b) Tanh Activation Function.

Figure 1.3: Continuous nonlinear Activation Functions of a simple Perceptron.

Finally, the Perceptron can show a linear behavior (Figure 1.4a), returning as output the exact value of the integration. Alternatively, the Rectified Linear Unit (ReLU) can be used, which returns the value only if positive (Figure 1.4b). This last one activation function is often used to introduce nonlinear behavior in the elaboration done by the neuron.



(a) Linear Activation Function. (b) ReLU Activation Function.

Figure 1.4: Linear and ReLU Activation Functions of a simple Perceptron.

1.1.2. Multi-Layer Perceptron

Starting from the Perceptron, multiple units can elaborate the information in series, composing a **Multi-Layer Perceptron**, which is the first form of a Neural Network (NN) [30]. Figure 1.5 represents an example of multiple neurons linked together in a cascade. Each neuron represents a layer, has its own activation function and elaborates successively the outputs of the previous ones. The neurons in the middle (N2 and N3 in the figure) are called Hidden Neurons, as their output is elaborated by the successive layer and is not directly accessible.

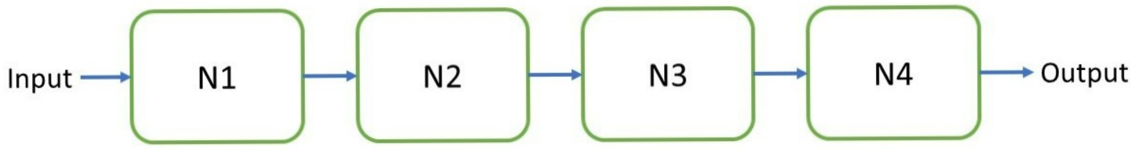


Figure 1.5: A stack of 4 perceptrons connected in series: N_1 is the input neuron, which receives the network input and introduces them in the network. The final neuron N_4 gives the output of the network, while the others are Hidden Neurons, as they return an output that is not directly visible.

1.1.3. Feed-forward Neural Network

Depending on the complexity of the problem that the network needs to solve, the different layers can be formed by more than one unit, and according to how the neurons are

connected among themselves, the final purpose of network itself changes. An example is the **Feed-forward Neural Network** (FFNN) shown in figure 1.6: the Input Layer is composed by 4 Input Neurons, as the input vector is made of 4 elements.
The successive Hidden Layer can modify these inputs in different ways according to the outcome needed: if the number of neurons is reduced with respect to the input, it will make an automatic selection of the inputs needed to correctly predict the output.
Finally, the output layer will have a number of neurons which depends on the purpose of the network. For instance, supposing that the network is used for the classification of three classes, at least three possible outcomes are needed. So if the output layer has neurons which return binary outputs, only one neuron is not sufficient to reach three final states: two neurons will be needed to produce at least 3 combinations. In fact, if only one neuron was used, the possible outcomes would be 0 or 1, which identify two cases. By adding another neuron, it is possible to create other combinations, up to 4 different cases: 00, 01, 10, or 11. Thus, two outputs are needed to identify 3 classes.

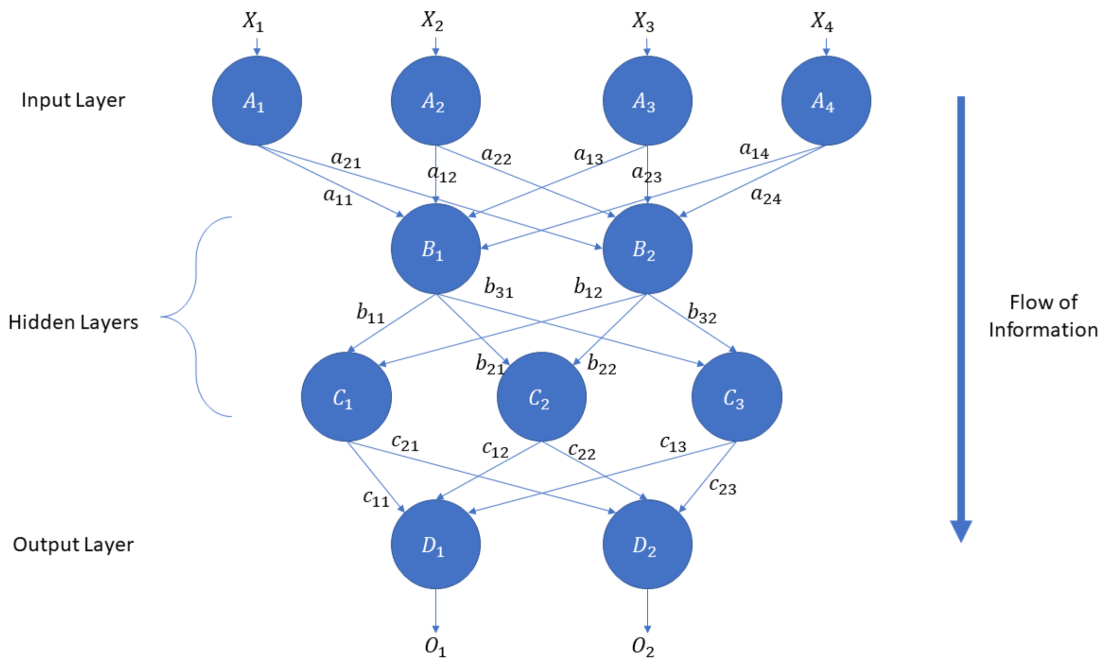


Figure 1.6: A 4 layers feed-forward Neural Network with 4 inputs and 2 outputs. The inputs are fed to the Input layer, composed by neurons A_1, A_2, A_3 and A_4, then follow a path which only goes forward to the successive layer. The output layer is composed by 2 neurons, whose activation functions will depend on the complexity of the problem to be solved.

In the FFNN the information flow is unidirectional, never going back [43]. The FFNN represented in Figure 1.6 is a **Fully Connected Neural Network** (FCNN), meaning that its neurons are always connected to every other neuron of the successive layer. This peculiar characteristic make it difficult to increase the number of layers over a certain number, as the computational cost becomes too high. Also, a FCNN can easily occur in the problem of overfitting, that is the loss of generalization of the model with respect to the data. If such thing happens, the network becomes incapable of performing correct predictions over data which are different from the ones used for its training.

1.1.4. Feedback Neural Networks

Another possible arrangement of neurons can be represented by **Feedback Neural Networks** (FBNN), where information flow can be reverted and go back in loops as input to other neurons of previous layers (see figure 1.7) [21].

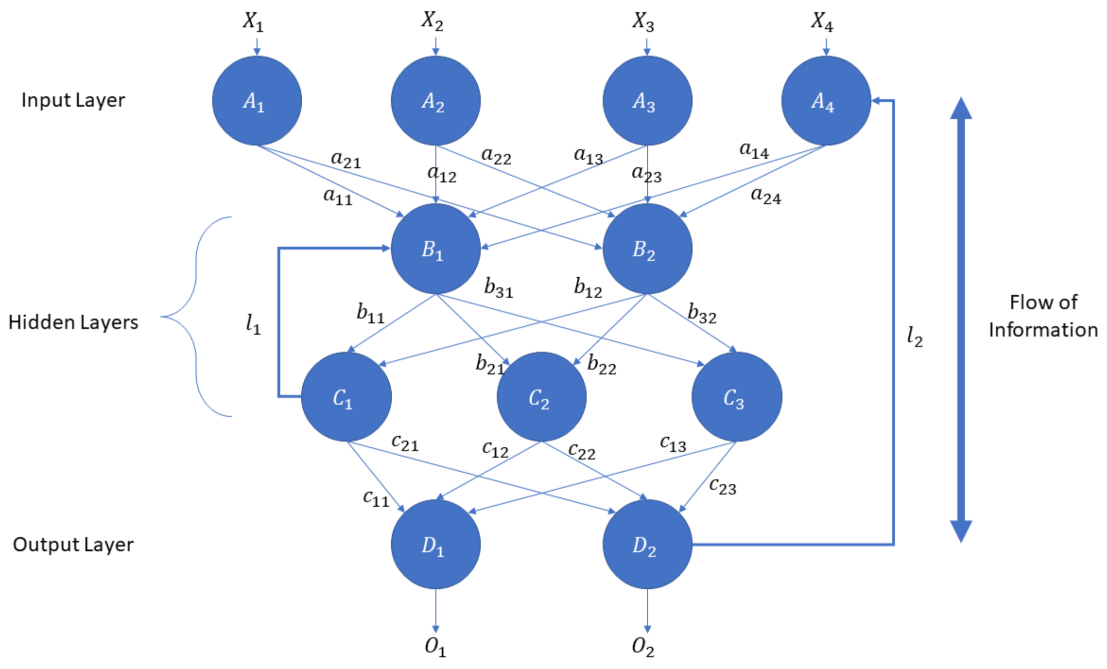


Figure 1.7: A 4 layers feedback Neural Network with 4 inputs and 2 outputs. The architecure is the same of the previous FFNN in figure 1.6, but two additional links are present, which make the output of some neurons to go back and influence the outcome of the elaboration.

In this type of networks time becomes an important element to take into account, due to the delay introduced with feedbacks: each discrete time step corresponds to a state of the network. Moreover, the involvement of outputs of previous time steps gives to these networks the ability to *remember* what happened before, making them suitable to recognize patterns in time. This is the reason why these architectures are employed to work with signals or, more generally, time series.

1.1.5. Recurrent Neural Networks

A particular case of FBNN are the **Recurrent Neural Networks** (RNN), in which the feedback loop may also occur over the same neuron, so that its own output becomes an input (figure 1.8). These types of networks may be employed as memory networks, thanks to their capability to store the state of neurons according to the input given [7].

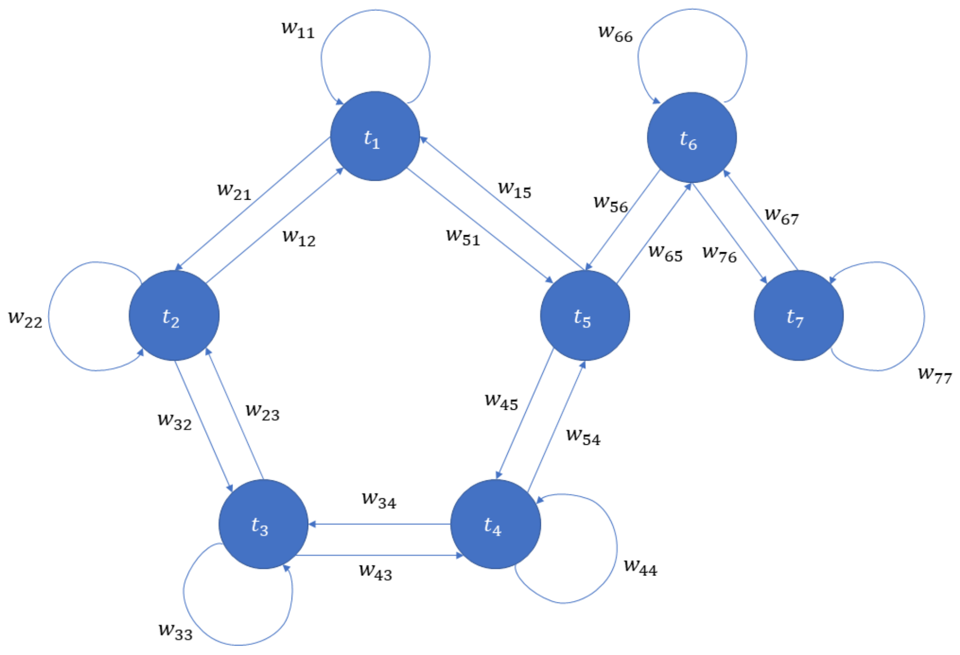


Figure 1.8: A Recurrent Neural Network, where the information is given to the various units in a series, through the corresponding weights w_{ij}, as well as to the same neurons with weights w_{ii}. These networks store a specific state given certain inputs, and can be used as memory networks, but also to perform predictions in time series.

1.1.6. Convolutional Neural Networks

As explained before, Neural Networks are inspired by the behavior of brain, thus it is important to understand how complex tasks are performed to be able to mimic them with an algorithm.

For instance, in the process of image recognition, the brain starts from the input, that is the image collected by the eye, and elaborates it to identify specific features. The elaboration corresponds to a gradual *abstraction* of the initial input: segmentation, local spatial relationships and modelling are possible operations that can help to picture the desired object.

Moreover, when the eye receptors acquire the frame containing a specific object, they performs a first degree of local elaboration through the so-called "receptive fields": the receptors of the image are subdivided in microareas that are smaller with respect to the whole picture, and each neuron focuses only on a part of it. These local receptive fields allow to understand the *local characteristics* of the object of interest, such as its contour, the directionality of its elements, and so on, helping in the identification of the aspects that portray an object.

The gradual abstraction and elaboration can also be seen as a *compression* of the information: once the final step is reached, the object is labelled with a specific word that embody all its essence; so with the word "glass" one can identify the shape, the local spatial relationships with the possible surrounding environment, the functionality and so on of that specific object. Figure 1.9 shows an example of possible compression performed by the brain on visual data.

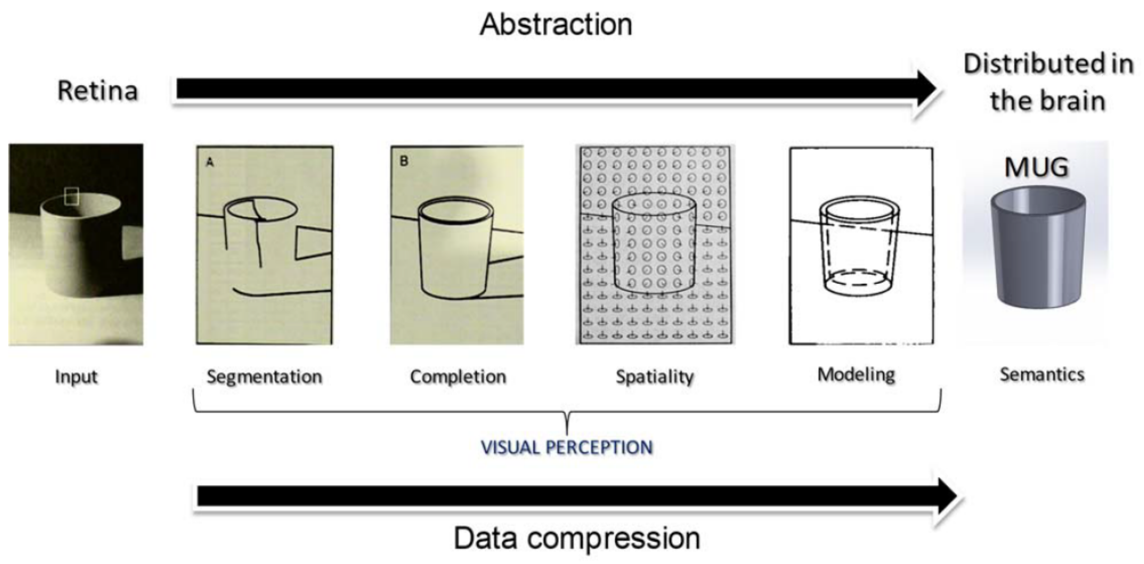


Figure 1.9: This schematic, taken from the notes shared by professor Cerveri of Lecture 4 of his part of the Neuroengineering course, shows the process of information elaboration followed by the brain, which starts in the retina up to the central brain cells, making a parallelism between data abstraction and its compression.

Placing these concepts in the framework of a Neural Network, the degree of elaboration and abstraction is represented by the *deepness* of the network, which is identified with the number of layers that manipulate the input. The deeper is a network, the more complex and unique the features extracted are. The other steps given by the behavior of our brain

are *local feature extraction* together with *data compression*.
Out of these conditions, FCNN may only partially achieve the last one by reducing the number of neurons of the successive layer. However, it is evident that this type of Network cannot be exploited, as its number of layers should be kept low to have reasonable computational costs, especially if multidimensional data are considered. Also, the full connection of one layer with its successive is opposed to the concept of locality of feature extraction, as in this way the network tries to find a relationship with every other point given as input.

To overcome this limitation, the **Convolutional Neural Networks** (CNN) have been developed [15]. They are Feed Forward Neural Networks which employ the concept of local receptive field in their architecture: each neuron of a layer is only connected to some neurons of the successive, the same way of eye receptors which acquire small areas of the image.

Another idea behind these Networks is weight sharing: when a local feature is extracted, the weights which identify that feature are fixed and repeated all over the layer, establishing a so called *feature map*. This means that a feature map corresponds to a specific feature that is extracted in the given input, and it will only have a number of parameters corresponding to the dimension of the local receptive field plus a bias.

These two concepts are represented in figure 1.10, which shows a simplified version of a feature map with a mono-dimensional input: given an input layer of 5 neurons, the connections with the successive hidden layer, which is a feature map, are not in the same number of received input, but consider only 3 inputs per neuron. Also, the 3 weights identified are kept and repeated in the feature map, so the neurons of the same layer *share* their weights.

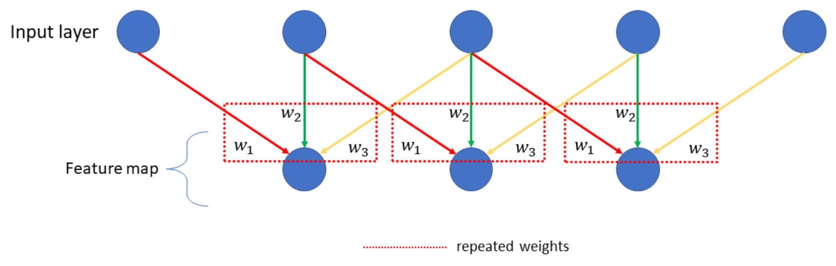


Figure 1.10: In this figure is shown that CNNs are not Fully Connected, and that feature maps are layers with shared weights, represented by repeating the same color sequence in the connections among neurons.

To obtain a feature map, a *convolution* of the input with a linear filter is performed. This operation allows to compress the data while extracting the feature. A common procedure is to identify different feature maps in the same layer, then proceed to the successive one and repeat the operations, until the final classifier is reached. This is possible thanks to the low complexity of feature maps in terms of parameters, that allows to extract numerous features and create a deep CNN with many layers.

For example, considering a Fully Connected Neural Network with an input layer of size 90x90 and the first hidden layer of size 80x80, the number of connections for that layer only would be 90x80, because every input neuron need to be connected to the successive neurons. Moreover, an additional 80 parameters, represented by the biases of each neuron, should be added. This brings to 7290 parameters that should be identified for a single layer, that represents a feature.
On the other hand, in a CNN with a convolutional filter of 5x5, the number of parameters for a single feature map would be only 25 weights + 1 bias. It is clear how the advantage in terms of computational cost allows to extract a lot of additional features with respect to the FCNN.

In Figure 1.11 is shown a graphical summary of how a convolution is performed on an image, obtaining an output known as feature map.

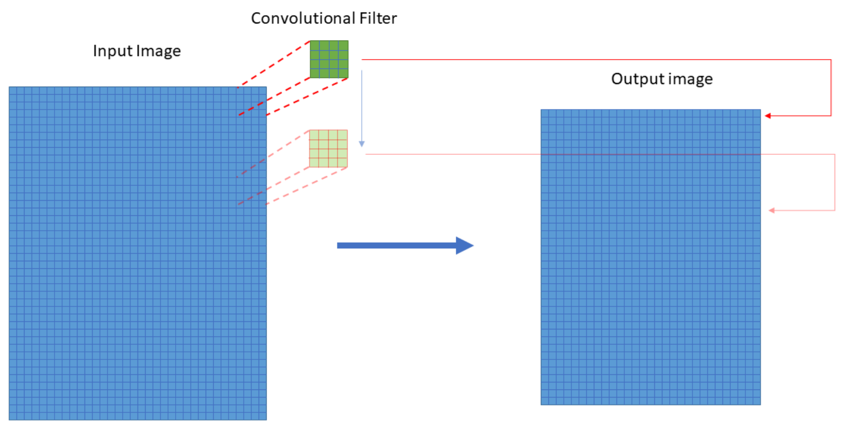


Figure 1.11: In this figure are shown the locality and data compression in convolution: given a multidimensional input, a linear filter is "slided" over the whole input. The operation returns an output value which summarizes the group of points used for the convolution with the filter. So the output image represents a compressed feature of the input extracted through the convolution.

Among all the architectures presented, CCNs are very important for the ECG classification, as these time series can be considered as multidimensional data thanks to their channels. By doing so, Convolutional Neural Networks may be exploited to automatically extract meaningful morphological features of these signals, which can be used for the classification of diseases [23].

1.1.7. Attention Mechanism

Another mechanism which takes inspiration from human information processing is the **attention mechanism**. Attention has not a clear definition, but can be seen as "the flexible control of limited computational resources" to "dynamically alter and route the flow of information", gaining benefits to adapt the system to its task [26]. Reporting it to brain image processing, it can be seen as the *focus* given over some details of the input received which are considered more important than others to understand the content of the picture considered.

In Artificial Neural Networks, the attention mechanism is performed using specific techniques that depend on the Network task: in image classification, an example can be represented by Class Activation Maps (CAMs), which are heatmaps that shows the most informative content of the image for the identification of the class. To obtain such maps, a Global Average Pooling (GAP) layer has been inserted in the last convolutional layer of the CNN [53], as shown in figure 1.12.

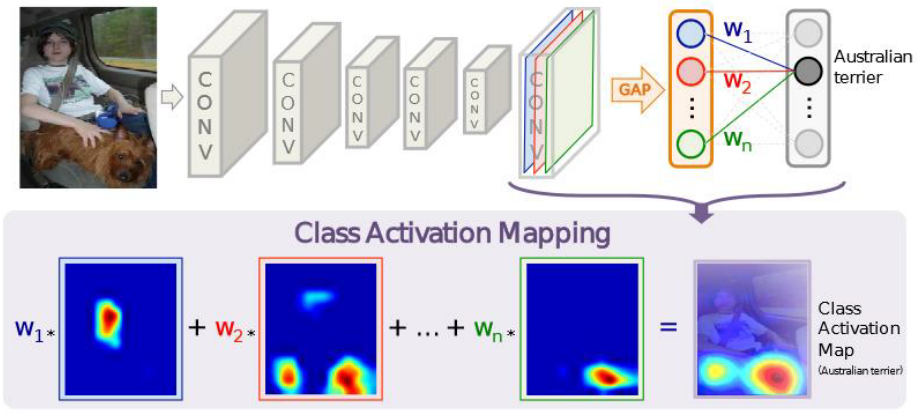


Figure 1.12: The figure, taken from [53], explains the concept of heatmap to focus the classification on details of the whole picture. As shown in the proposed architecture, the GAP is placed just before the final classification layer.

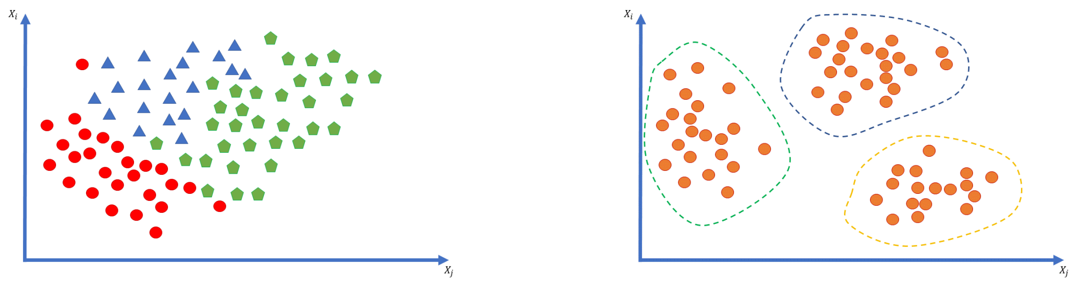
Attention mechanism is a very important tool to improve classification performances, as it allows to mantain the most informative content of the input only, thus reducing errors.

1.2. Network training

Having seen the possible structures in Neural Networks, in this section it will be illustrated how their training is performed. As for the architectures, also the learning process of Neural Networks gets inspiration from brain neurons. In order to modify its connections, and change their effects on the structure of brain, a neuron needs external stimuli, which can modify the behavior of a synapses, making them excitatory or inhibitory. Moreover, the neural cells may generate or remove pathways through which the electric impulses are sent, depending on the fact that the way is practiced or not. These stimuli are external factors to which a person needs to adapt, thus specialize toward an ability, or in other words *learn* a function [29].

In the context of a Neural Network, the concept of **learning** is put into practice as the computation of weights and thresholds that characterize each neuron. Opposed to the brain, an algorithm cannot change the organizational structure of its neurons, but it can still modify the behavior of its connections by giving them more or less importance, and by making them positive (excitatory) or negative (inhibitory). The stimuli exploited to modify the network parameters are the training data, through which the network learns to map the data features, or predictors, for a specific task. Thus, training an algorithm means feeding it with the training data to predict an output that depends on the chosen task.

The tasks that can be learnt may be of two different kinds: **supervised** or **unsupervised**. The former is performed whenever the training data presents both the predictors and the desired output. For instance, in classification task, data are given together with their classes, and the network will learn to recognize which characteristics are typical of each output (see figure 1.13a for an example).
In the second one, the objective is to find similiarities or patterns among input data without knowing an a-priori desired output. An example is depicted in figure 1.13b, where the clustering task is performed: without knowing a grouping criterion, the labels are crafted automatically by the network itself during the training, so that they can be identified with a class according to their characteristics. After the procedure, the network will place the data in clusters.



(a) Supervised problem - Classification. (b) Unsupervised problem - Clustering.

Figure 1.13: In figure 1.13a is shown classification, where the training data are a-priori divided in known classes - triangles, circles and squares. Using the predictors X_i and X_j the algorithm should learn to assign a class to unlabelled data. Figure 1.13b shows clustering, an unsupervised task where the points have no label - all circles - but similiar subsets can be found. The algorithm learns the clustering criterion and creates the grouping by itself.

1.2.1. Learning rules

This work will focus on classification, which is a **supervised learning** problem, so the network is trained on data whose label is known.

In supervised learning, parameter tuning happens through the computation of an error: if the network predicts correctly, no changes will be needed; however, if the predicted label is not correct it means that the parameters are to be modified, and the error will be used as a correction factor: the bigger the error, the bigger the correction applied on the network parameters will be.

For error computation and weight correction, the technique implemented in multi-layer networks is the error backpropagation, which is an extension of the Delta rule used for single layer networks.

Delta rule is a gradient descent method used in neurons with continuous activation functions: given an input and knowing the wanted output, the squared error E is computed and used to correct the weights proportionally. The gradient descent method is chosen because if the error increases with an increase of the weights, these parameters need to be reduced, so they need to go in the *opposite* direction of the error.

This iterative method can be performed as long as an error can be computed. If the network is made by a single layer, its output can be compared directly with the expected label, calculating the error. But as soon as other hidden layers are added to the network, it will be impossible to know their expected output, thus the hidden parameters cannot be updated.

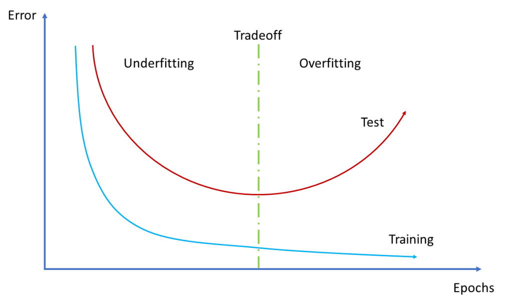
The solution for the computation of error in hidden layers is error **backpropagation**, which exploits the error of the output layer to compute the one in all the previous hidden layers [43]. To do so two steps are followed: first, the *forward step* expresses the outputs of every neuron in the various hidden layers up to the final output layer. Then, a *backward step* is applied, which takes into account the error of the output layer and takes it back in the previous hidden layer exploiting the derivative of the activation function of the neuron considered. Error backpropagation allows to train multi-layer networks, which are needed to perform complex operations.

1.2.2. Overfitting

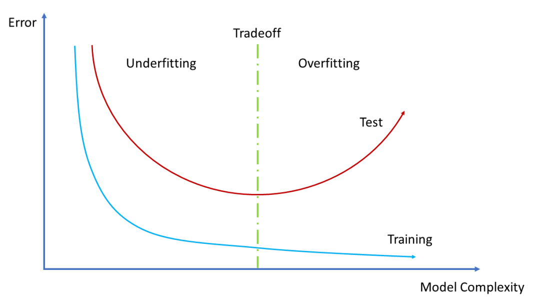
The main problem in model training is to guarantee the correct *generalization*, that is the capability to correctly predict the output of data which do not belong to the training dataset. The two learning rules shown are *iterative*, meaning that they are repeated over all the dataset more than once to reduce the loss function as much as possible. Each iteration over the dataset is called **epoch**. Usually, at each new epoch the error is reduced, but the model will also be inclined toward a loss of generalization, as it focuses too much on the training data. Thus, a trade-off between the number of epochs and generalization should be found to have good predictions over unseen data (see figure 1.14a).

To find this optimal point, the metrics computed on the training data are not a good index of how the Network will perform on unseen recordings. In fact, during the training over a dataset, the model fits its parameters to identify those specific data. However, if the model learns to classify only those specific data, it may not be able to predict correctly different data, causing the so called **overfitting**.

For this reason, the model performance evaluation occurs over an unseen group of data, which is called **test set**. This selection of data is obtained by keeping a part of the initial training dataset hidden to the model. The separation of training and test data must be done before any operation on the data, else the model performances would be influenced. Overfitting may also occur due to an excessive or insufficient complexity of the network architecture, so another trade-off between model complexity and generalization capability of a Network has to be identified, as shown in figure 1.14b.



(a) Tradeoff Error - Epochs.



(b) Tradeoff Error - Model Complexity.

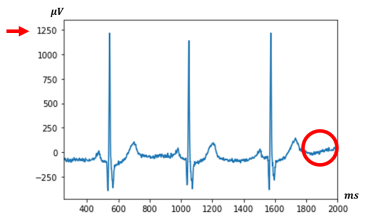
Figure 1.14: The graph 1.14b shows the evolution of the error performed in the classification of data with the increase of the chosen complexity of model: as the latter increases, the model will fit better to all the points in the training dataset, resulting in a decrease of the former, represented with the blue line. However, the model will also lose generalization, represented by the increased error performed in the predictions over the test set, represented with the red line. The optimal point is represented by the minimum identified in the test set error curve, else it would occur in underfitting or overfitting. The same considerations can be done observing graph 1.14a, watching how the epochs change the goodness of the model.

To improve model generalization some operations can be performed. In particular one can act with data processing, over the model architecture and on the training procedure:

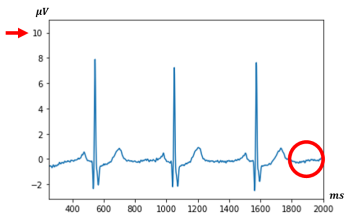
- **Data processing** can be used to filter, standardize and remove biases, so that differences in data coming from different sources can be nullified. For example, if a model is trained with data which have been acquired with different protocols, a filtering and a standardization may remove the differences in bias and noise affecting the signals.

  Taking as example ECG signals, in figure 1.15a is shown a raw recording where both the scale and the oscillation in signal are high. After a filtering and z-normalization procedure (figure 1.15b), the obtained time series has been scaled down and became more stable.

  Data processing is of primary importance for the correct training of a model, because it enhances the quality of data and ensures that the model inputs have the same shape, reducing the possible mistakes in learning.



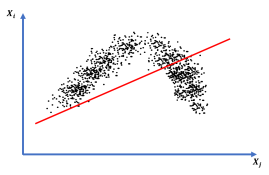
(a) Raw ECG signal.



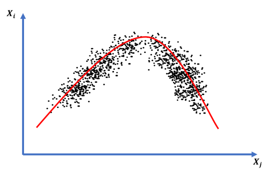
(b) Processed ECG signal.

Figure 1.15: Considering different sources, the scales may vary according to each center of acquisition due to differences in hardware and sampling frequencies, thus it is very important to follow a procedure which brings all the signals on the same level before feeding them to the model.

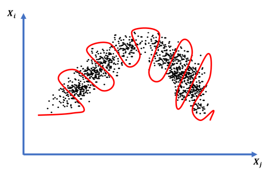
- **Model architecture** can be changed according to the degree of complexity needed for the correct characterization of the problem. In fact, if the network fits to training data perfectly but performs poorly on test set, it may be due to the excessive complexity given to the network itself.
  For example, if the points are distributed over a parabolic line, the model needs to map a quadratic relationship; but if we feed it to a network with a lot of hidden layers, it may create a complex nonlinear relationship that is far from the wanted distribution, and predictions over unseen data will be all wrong (see figure 1.16c).
  The same thing may happen in the other direction, so also if the model is too simple it may not generalize well, causing underfitting, as shown in figure 1.16a.
  For this reason, sometimes it can be useful to modify the number of layers, or to change the complexity of model activation functions, to prevent the two phenomena.



(a) Underfitting.



(b) Correct Modelization.



(c) Overfitting.

Figure 1.16: The three figures explain how important is to identify the optimal model without occurring in underfitting (figure 1.16a) or overfitting (figure 1.16c): if training occurs without the right complexity, the predictions (represented by the **red line**) would not follow the actual behavior of data, giving misleading outcomes for unseen data.

Also, *Dropout layers* can be added in the architecture, which randomly sets the input of the next layer to 0, making more difficult for the model to overfit on the training data and correcting possible mistakes of previous layers.

- **Training procedure** can be also affecting the overfitting, and can be modified to avoid it and improve generalization. First of all, some *regularization* terms can be added to the loss function, making it penalize large weights, which are often cause of overfitting.
  Also, early stopping can be performed during training: after fixing the number of epochs of training, it can be stopped before according to the performances reached if they are above or below a certain threshold. For instance, if the loss value variation is too small between two epochs it means that the model is starting to overfit, thus its training is not effective anymore and further epochs would only worsen its generalization capability.
  To better understand the generalization capability of the model *during* training, the **validation set** can be employed. This set of data is obtained by further splitting the training data, as done with the test set, and hidden to the model during training: after each epoch, an evaluation over the validation set is performed. The metrics obtained from this hidden group can be used as reference for the early stopping to have a more reliable index of the generalization capabilities in itinere. This way, as soon as the model starts to overfit over the training dataset, the training procedure can be stopped.

1.2.3. Ensemble of models

A methodology employed to improve the generalization of a model in the final predictions is the **ensemble** of models [37]. This technique is performed by training different networks which are then used together to perform the same prediction, as shown in figure 1.17.
There are different methodologies to integrate the outputs of single models to obtain a final result. One way can be *majority voting*, so the class assigned is the one selected by the majority of models. As the models are different from each other, the error on test set will also change, making different decisions, thus generalizing better.
However, to have different models, one also needs to train them on different data, thus the ensemble is usually exploited when a large and heterogeneous dataset is available.

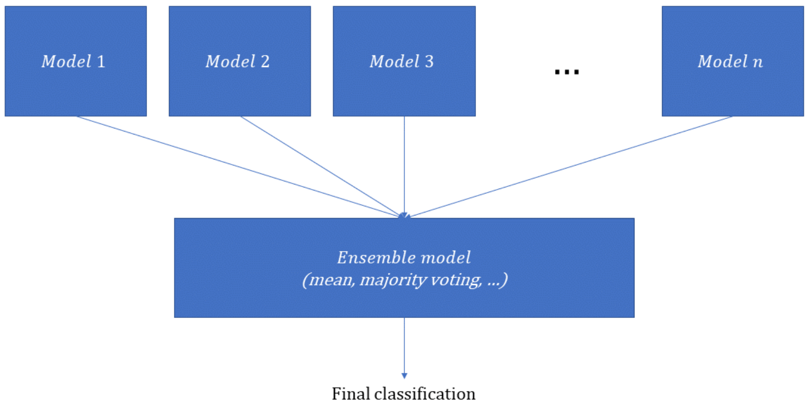


Figure 1.17: Scheme of the ensemble of n models, where each model computes its prediction, contributing to the production of the final output.

1.3. Neural Networks for ECG classification

In the context of ECG classification, several Deep Learning algorithms have been presented in literature, from simple FCNN, passing through the more structured Long-Short Term Memory (LSTM) Networks, which are a type of RNNs that exploit the time series nature of the recordings, up to the CNNs, which take into account relationships among different portions of the signal [23].
However, these algorithms have been trained and evaluated on small and homogeneous datasets, learning the classification of a small number of CVDs. Thus, their success cannot be considered meaningful for the employment in real cardiological diagnosis, as it does not represent the complexity of ECG interpretation [39].

The PhysioNet/Computing in Cardiology Challenge 2020 takes place in this setting to address this problem. The participants were asked to come up with an automatic ECG classifier which has to be trained over a vast dataset coming from several acquisition centers, with each recording associated to a large number of cardiac abnormalities. Moreover, the 2021 Challenge posed a further objective, asking to classify the ECG with reduced lead sets: not only the complete 12-leads, but also 6, 4, 3 and 2-leads subsets classifiers were required. This was done to understand if the algorithms were able to extract meaningful features within single channels, and whether they can be enough to understand the abnormalities affecting the subjects [40].

The work presented in this thesis started from the first and second classified in the 2020

Challenge [34, 50] and Federico Muscato's Master Thesis [33] to take part in the 2021 Challenge.

The main idea behind the **prna** team, which arrived 1st in the 2020 Challenge, was to employ handcrafted features together with a Neural Network in the classification, merging together Machine Learning and Deep Learning approaches. This was done to improve the performances of the model with respect to the use of a Network alone, as well as to create an explainable interpretation of the classification. The signals followed two parallel pathways before reaching the final classifier, called "Wide" and "Deep" branches.
In the **Wide** branch the handcrafted features were extracted from the ECGs. These features were concatenated to the outputs of the other branch before being fed to the classifier layer. In the **Deep** branch the actual Neural Network was implemented. This part of the model was responsible to automatically extract the implicit signal features used for the prediction. As said before, the outputs of the two branches were concatenated and used for the final prediction of the 24 classes of the Challenge.

On the other hand, the team **between a ROC and a Heart place**, ranked 2nd in the 2020 Challenge, developed a Residual Network, which is a Convolutional Neural Network, with attention mechanism implemented through Squeeze-and-Excitation (SE) blocks. Considering the multiple channels recorded in each ECG, the network input could be considered bidimensional. This allowed to use the Residual Network, typically exploited in image recognition, to find relationships among the portions of the signal in a single channel. The attention mechanism represented by the SE blocks allowed to catch spatial channel interdependencies. Results obtained show how this type of network successfully catches the *morphological* features which characterize each cardiac abnormality.

Federico Muscato's Master Thesis exploited this last cited work's network and focused on the issue of data unbalancing which characterizes the datasets considered: in fact, the data given by the Challenge presented unbalanced classes, with some predominant labels which outnumbered by far other rarer ones. To overcome this limitation, he put his attention on rebalancing and generalizing the model as much as possible in training: first of all, he identified the most numerous class, represented by the Normal Sinus Rhythm (NSR). Then he divided the number of recordings which was labelled with this class in 3 subsets and trained three models with the same architecture using all the dataset, but a different third of NSR. Finally, the predictions were performed by an ensemble of the three models, which assigned the class by majority voting. The **ensemble** was used to improve generalization of predictions.
Another step implemented was a **threshold optimization** process performed on the final classifier layer. The procedure found the optimal thresholds for the predictions, reducing

or increasing them according to the Challenge Metric computed at the end of training. This measure contributed to the reduction of class unbalancing because, if a label is more represented than another during training, the algorithm predicts the frequent class more often than the other, sometimes even neglecting the uncommon ones. Thus, in order to improve the prediction of the less represented targets, the procedure chose the correct threshold for the classification. Both the methodologies contributed to improve the results of the previous work.

This thesis work starts from the background given by these works taking inspiration by three aspects:

- From the winner of the 2020 Challenge was implemented the integration between Machine Learning and Deep Learning approaches, trying to identify the *rhythmic* and *morphological* features that characterize each CVD taken into account in the Challenge.
- The ResNet with Attention Mechanism of Zhao et al. was taken as basis of the Deep branch network, and further expanded with Dilation Layers in the convolution to improve generalization of the extracted features.
- The ensemble classification and threshold optimization methodologies were taken from Muscato's work to deal with unbalancing of the dataset.

Developing these three points, an algorithm was prepared to take part to the 2021 Physionet Challenge. The model was trained on the publicly available data and tested in the Official Phase over the hidden test set. After the Challenge, other changes were introduced, and the performances were evaluated over a stratified local test set kept hidden during the training of the modified network.

2 | Materials and Methods

This chapter will go through the available data used in this work, and their description with a focus on the problem of unbalancing. Moreover, a discussion over the labelling criterion will be done, to understand some of the issues met during the Challenge. Then, the data processing steps will be shown. Finally, the model architecture and the training procedure will be described in detail, together with the evaluation methodology which gave the final results presented in Chapter 3.

2.1. Data description

The PhysioNet/CinC Challenge 2021 main topic was the ECG classification, so the nature of this signal will be first presented.

2.1.1. Electrocardiogram

The electrocardiogram (ECG) is the recording of the electrical activity of the heart extracted through the use of various electrodes, whose position over the body has been studied to obtain the most informative content about the heart's condition [4, 32, 42]. It is a quasi-periodic signal whose amplitude is usually in the order of mV. Its morphology is typically composed by 5 peaks, called fiducial points, appointed with the letters P, Q, R, S and T.

The fiducial points identify 3 informative regions in the ECG: the P wave, the QRS complex and the T wave. The first one regards the atrial depolarization event, during which this part of the heart starts to contract. Then the QRS complex happens, which is the manifestation of the occurrence of ventricular contraction. Usually the QRS is the most prominent peak seen in the ECG recording, and it is superimposed to the wave that represents the atrial repolarization. For this reason, it cannot be noticed in normal conditions [22]. Finally, the T wave represents the ventricular repolarization.

By taking into account two or more points, peculiar segments can be identified: for example, the PR interval represents the time between atrial and ventricular depolarization,

which are the electrical events that identify the muscular contraction of the heart chambers. In other words, the PR interval is proportional to the velocity of the electric signal travelling from atrium to ventricle [11].
Another characteristic interval is the QT interval, which represents the time between contraction and relaxation of heart ventricular muscle [10]. According to the duration, the amplitude and the morphology of the waves, the segments and the complex, cardiologists can determine the health or pathological status of the subject's heart.

In figure 2.1 is shown the typical morphology of a single beat of the ECG in a healthy patient, with all the fiducial points and segments highlighted.

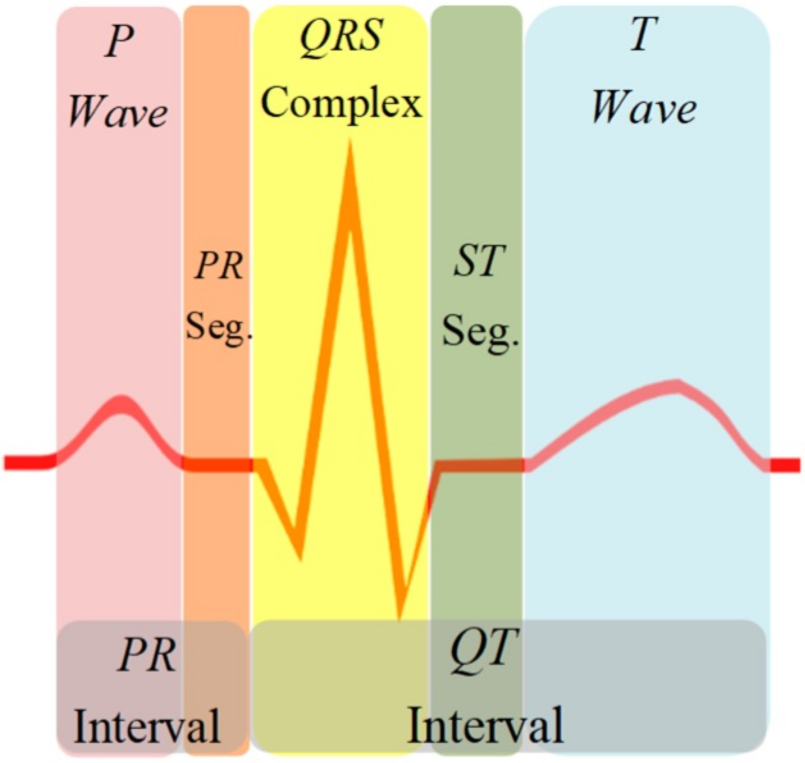


Figure 2.1: This figure, taken from [4], represents the template of an heartbeat in an electrocardiographic recording.

The standard clinical acquisition of an ECG is a 12-leads recording, which represents a 12 channels signal acquired with the use of 10 electrodes [42]. There are 4 standard electrodes, each placed over one human limb to follow the *Einthoven's Triangle* configuration [13]. Out of these four electrodes, one is used as reference and the other three record the variation of potential of the heart dipole. From the recordings obtained, 6 leads are extracted: lead I, II, III, or standard limb leads, and aVL, aVR, aVF, which are the augmented limb leads [13, 14]. These six leads describe the heart activity projection on

the frontal plane [42].
The other 6 electrodes are placed over the patient's chest, close to the heart. The obtained leads are called precordial leads, and their names are V1, V2, V3, V4, V5 and V6. They describe the projection of the heart dipole on the trasversal plane [5].

Figure 2.2 shows a scheme of ECG clinical recording, representing the approximate position of the electrodes (without considering the reference electrode of the left leg), and how the heart electrical dipoles represented by the acquired leads are oriented in space.

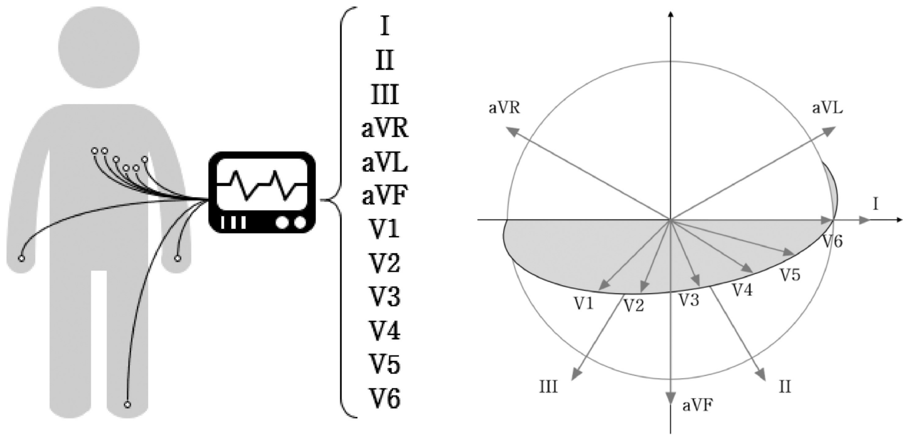


Figure 2.2: A scheme of the position of electrodes for ECG acquisition and the space distribution of leads, taken from [38].

Different factors can influence the acquisition of an ECG: electrode-skin impedance problems, as well as hardware configuration and environment conditions, can create biases or changes in the recording. In fact, from person-to-person the skin characteristics may vary, and the impedance given by the subject can be slightly different, changing the signal visualized. Also, American power line oscillates at 60Hz, while in most European countries the power-line frequency is 50Hz, thus the noise superimposition changes frequency according to the country of origin. Moreover, if a different hardware configuration is used, the characteristics of the signal can change.

This is the reason why working with data which come from different sources and acquisition centers can be difficult, and this can bring the necessity to standardize and normalize data before using them for any elaboration: in particular, during the PhysioNet/CinC Challenge this effort was required to face the multi-source dataset provided.

2.1.2. PhysioNet/CinC Challenge Dataset

As just mentioned, the PhysioNet/CinC Challenge 2021 made available a vast dataset composed of 88253 recordings labelled with 133 different cardiac abnormalities. Specifically, 7 public databases were the sources of these recordings. They are:

1. The China Physiological Signal Challenge (CPSC) Database and CPSC-Extra Database, which together contained 13256 signals [27].
   Out of these data, only 10330 (distributed as 6877 and 3453 recordings from each dataset respectively) were shared to the teams as training data, while the remaining 2926 were kept hidden as validation and test for the evaluation of the submissions. The recordings coming from these datasets were sampled at 500 Hz, and had a variable length between 6 and 144 seconds.

2. The Institute of Cardiological Technics (INCART) Database from St. Petersburg, Russia was made up of 75 annotated long recordings [46]. All the signals were 30 minutes long with a sampling frequency of 257 Hz, and were all given as training data.

3. The Physikalisch-Technische Bundesanstalt (PTB) Database from Berlin, Germany [9, 47]. It was composed by the PTB and PTB-XL datasets: the first one contained 549 ECGs from 290 patients, while the latter had 21837 clinical recordings from 18885 subjects, for a total of 22,353 signals.
   Each recording had a length ranging from 10 to 120 seconds with a sampling frequency between 500 and 1000 Hz. All data were shared as training data.

4. The Georgia 12-leads ECGs Challenge Database, collected in the southeastern state of the United States of America, contained 20672 recordings. The signals' length was between 5 and 10 seconds, sampled at 500 Hz. Only 10344 ECGs were given as training data, while the remaining 10328 were kept hidden as validation and test set.

5. The Chapman University (California, USA) collaborated with Shaoxing People's Hospital (Chapman-Shaoxing) to create a large public dataset of ECGs [52]. Also, Ningbo First Hospital (Ningbo) shared other recordings, all coming from Zhejiang, China [51]. These sources provided together 45,152 ECGs, all with a duration of 10 seconds and a sampling frequency of 500 Hz, shared as training data.

6. An hidden American database, whose geographic provenance differs from the Georgia database, contained 10,000 ECGs, all retained as test data, so their nature and characteristics are unknown.

7. The last database comes from University of Michigan (UMich), containing 19,642 ECGs all kept as test set. Their characteristic length is 10 seconds and their sampling frequency is either 250 or 500 Hz.

During the Challenge, the performances of the classifiers presented by the participants were evaluated only on 30 out of the available 133 pathologies [40]. Moreover, among the 30 CVDs, there were 4 couples of diseases which were considered equivalent for their similiar effect on the ECG. This equivalence was expressed by rewarding the couples of pathologies with the same score during the classification. Specifically, there were:

- Complete Left Bundle Branch Block (CLBBB) scored as Left Bundle Branch Block (LBBB);
- Complete Right Bundle Branch Block (CRBBB) scored as Right Bundle Branch Block (RBBB);
- Premature Atrial Contraction (PAC) scored as Supraventricular Premature Beats (SVPB);
- Premature Ventricular Contraction (PVC) scored as Ventricular Premature Beats (VPB).

For this reason, the classifiers could focus on the prediction of only 26 classes instead of the original 30, considering the couples as single labels. In Table 2.1 is presented the number of labels in each training dataset and the sum of the instances of such classes among the various databases. The equivalent labels are highlighted with the same colour.

Labels	CPSC	CPSC-Extra	StPetersburg	PTB	PTB-XL	Georgia	Chapman-Shaoxing	Ningbo	Total
AF	1221	153	2	15	1514	570	1780	0	5255
AFL	0	54	0	1	73	186	445	7615	8374
BBB	0	0	1	20	0	116	0	385	522
Brady	0	271	11	0	0	6	0	7	295
CLBBB	0	0	0	0	0	0	0	213	213
CRBBB	0	113	0	0	542	28	0	1096	1779
IAVB	722	106	0	0	797	769	247	893	3534
IRBBB	0	86	0	0	1118	407	0	246	1857
LAD	0	0	0	0	5146	940	382	1163	7631
LAnFB	0	0	0	0	1626	180	0	380	2186
LBBB	236	38	0	0	536	231	205	35	1281
LQRSV	0	0	0	0	182	374	249	794	1599
NSIVCB	0	4	1	0	789	203	235	536	1768
NSR	918	4	0	80	18092	1752	1826	6299	28971
PAC	616	73	3	0	398	639	258	1054	3041
PR	0	3	0	0	296	0	0	1182	1481
PRWP	0	0	0	0	0	0	0	638	638
PVC	0	188	0	0	0	0	0	1091	1279
LPR	0	0	0	0	340	0	12	40	392
LQT	0	4	0	0	118	1391	57	337	1907
QAb	0	1	0	0	548	464	235	828	2076
RAD	0	1	0	0	343	83	215	638	1280
RBBB	1857	1	2	0	0	542	454	195	3051
SA	0	11	2	0	772	455	0	2550	3790
SB	0	45	0	0	637	1677	3889	12670	18918
STach	0	303	11	1	826	1261	1568	5687	9657
SVPB	0	53	4	0	157	1	0	9	224
TAb	0	22	0	0	2345	2306	1876	5167	11716
TInv	0	5	1	0	294	812	157	2720	3989
VPB	0	8	0	0	0	357	294	0	659

Table 2.1: Label distribution per dataset officially provided by the PhysioNet/CinC Challenge 2021 [40].

Inhomogeneity in labelling criterion

One of the issues coming from the multiple-source nature of the PhysioNet Challenge dataset was the **non-homogeneous** classification criterion for the pathologies: in fact, from table 2.1, it can be noticed that some of the datasets, such as the CPSC, the StPetersburg or the PTB, did not present all the labels.
This could be due to the fact that, during the preparation of those databases, certain diseases were neglected. Also, there is the possibility that some grouping have been performed: for instance, in the Ningbo dataset the Atrial Flutter (AFL) appears in 7615

samples, while the Atrial Fibrillation (AF) never occurred. Being the former a rarer CVD with respect to the latter, probably they were grouped together due to their similarity.

Data selection

This difference in labelling criterion among the datasets brought a problem in training the algorithm, because it supposedly created a bias towards some pathologies, neglecting some others. In fact, if a pathology was present in the record, but the ECG was not labelled with it, it would not be possible to train a model to recognize it. Moreover, if in some dataset the label is present, while in others is not, this would mislead the network's training phase, not allowing it to correctly understand the CVD's features.

For this reason, a dataset selection was performed before introducing the samples in the network to maximize the performances of the algorithm as well as its generalization capability in training.
First of all, the StPetersburg INCART database was excluded for the duration of its signals, which were far longer (30 minutes) with respect to all the other data, whose length was in the order of a few seconds. The difference in length could have been troublesome for the model training: in fact, the input size of the model would have not allowed to use the whole duration of such signals, thus it would have been necessary to take partial time windows. Using time windows could be misleading for the training of the network, because if the recording was labelled with a pathology which occurs in a specific part of the track, only some segments of the considered signal would actually show it. Thus, the model would learn to classify wrongly the pathology if the considered segment did not present it. The PTB dataset was also kept out because of the reduced population size from which the data were extracted: in fact, the 549 recordings were obtained from 290 subjects only. Moreover, the number of the Challenge pathologies contained was very small with respect to other datasets.
Finally, the CPSC-Extra was not considered for the poor quality of its data.

After the exclusion of these databases, the rejected data were 4077, thus the dataset reached 84176 recordings. A further data selection procedure was then applied to make the signals used as homogeneous as possible: first of all, the ECGs which did not present any of the Challenge labels were removed, then only 10-seconds long signals sampled at 500 Hz were included in the final dataset, resulting in 67659 recordings. Finally, a stratified holdout of 9646 samples was performed to create a local test set upon which evaluate the performances of the algorithm after the training.

2.1.3. Cardiac abnormalities in PhysioNet Challenge

Among the various labels assigned to each ECG, the Normal Sinus Rhythm (NSR) is the control class which shows a regular healthy heartbeat. Any variation from it is considered a cardiac abnormality, and the alteration caused on the recording can be **rhythmic**, **morphological** or both.
Specifically, if the alteration is focused on the rhythm, it means that the heartbeat track is almost normal, but it pulses at frequencies which are not in the normal range: this is the case of Sinus Bradycardia (SB) or Sinus Tachycardia (STach)[18, 20]. There are also other variations, like Prolonged PR Interval (LPR), which modify the duration of specific segments of the ECG [3].
On the other hand, if a CVD is focused on the morphological alteration of the heartbeat, it will mean that there is a conduction alteration in the heart cells which divert the usual pathway of the electric dipole. Thus, the ECG recording will be altered from the template track shown in figure 2.1. This is the case of Bundle Branch Block (BBB), where the QRS complex shape is modified; also the Q wave abnormality (QAb) or the Left Axis Deviation (LAD) show an alteration of specific parts of the ECG [24, 28, 31].
Naturally, the subtle line which separates the two natures of CVDs makes it difficult to distinguish them clearly: in fact, some temporal alterations can bring changes to the morphology of the ECG, but also the other way round is valid. For instance, Atrial Fibrillation (AF), which shown no P wave and an "irregularly irregular" pattern together with a fast rhythm [36], has a double nature of being both a rhythmic and morphological alteration.

Table 2.2 summarizes the hypothesis elaborated over each pathology, after an accurate analysis of the diseases' definitions given on the PubMed website [1], to classify their effect on the ECG.

Labels		Category	Observations
AF	Atrial Fibrillation	Both	Irregular rhythm for long time
AFL	Atrial Flutter	Both	Flutter waves and fast rhythm
BBB	Bundle Branch Block	Morph	Affects shape of QRS complex
Brady	Bradycardia	Time	Slow pacing rhythm
CLBBB | LBBB	(Complete) Left Bundle Branch Block	Morph	Widened QRS complex
CRBBB | RBBB	(Complete) Right Bundle Branch Block	Morph	Widened QRS complex
IAVB	1st deg atrioventricular block	Time	Prolonged PR Interval
IRBBB	Incomplete Right Bundle Branch Block	Morph	Widened QRS complex
LAnFB	Left Anterior Fascicular Block	Morph	Alterated QRS complex
LAD	Left Axis Deviation	Morph	Shift of QRS orientation
LQRSV	Low QRS Voltages	Morph	Altered QRS amplitude
NSIVCB	Nonspec. Intraventr. Conduct. Disorder	Morph	Altered QRS complex
NSR	**Normal Sinus rhythm**	Both	Normal behavior
PAC	Premature Atrial Contraction	Both	Premature P wave, no QRS
PR	Pacing Rhythm	Morph	Altered shape with PM[1]
PRWP	Poor R wave Progression	Morph	Altered V1 to V6 R-peak
PVC	Premature Ventricular Contraction	Morph	Altered and wide QRS complex
LPR	Prolonged PR Interval	Time	Longer time btw P and R peak
LQT	Prolonged QT Interval	Time	Longer time btw Q and T-wave
QAb	Q Wave Abnormal	Morph	Altered Q-wave shape
RAD	Right Axis Deviation	Morph	Shift of QRS orientation
SA	Sinus Arrhythmia	Time	Irregular heartbeats
SB	Sinus Bradycardia	Time	Slow pacing rhythm
STach	Sinus Tachycardia	Time	Accelerated pacing rhythm
SVPB	Supraventricular Premature Beats	Both	**Same as PAC**
TAb	T Wave Abnormal	Morph	Altered T-wave shape
TInv	T Wave inversion	Morph	Shift of T-wave orientation
VPB	Ventricular Premature Beats	Morph	**Same as PVC**

1. Pacemaker.

Table 2.2: Here is listed the label division in three groups of ECG alteration: **morphological** as green, **temporal** as light blue or **both** as yellow. NSR is considered the control class for both the alterations.

These considerations were taken into account to build the model in its two branches fashion: the idea was that one branch, called Wide branch, would focus on recognizing the temporal characteristics of the recordings, while the other one, the Deep branch, had to put its attention over the morphology of the signal, improving the performances with respect to the use of a single network.

2.2. Data processing

A separate data processing was designed to extract two distinct inputs to be fed to each branch, making the same recording to follow a double pathway. Figure 2.3 shows a summary of the operations conducted over the same signal for the two pathways.

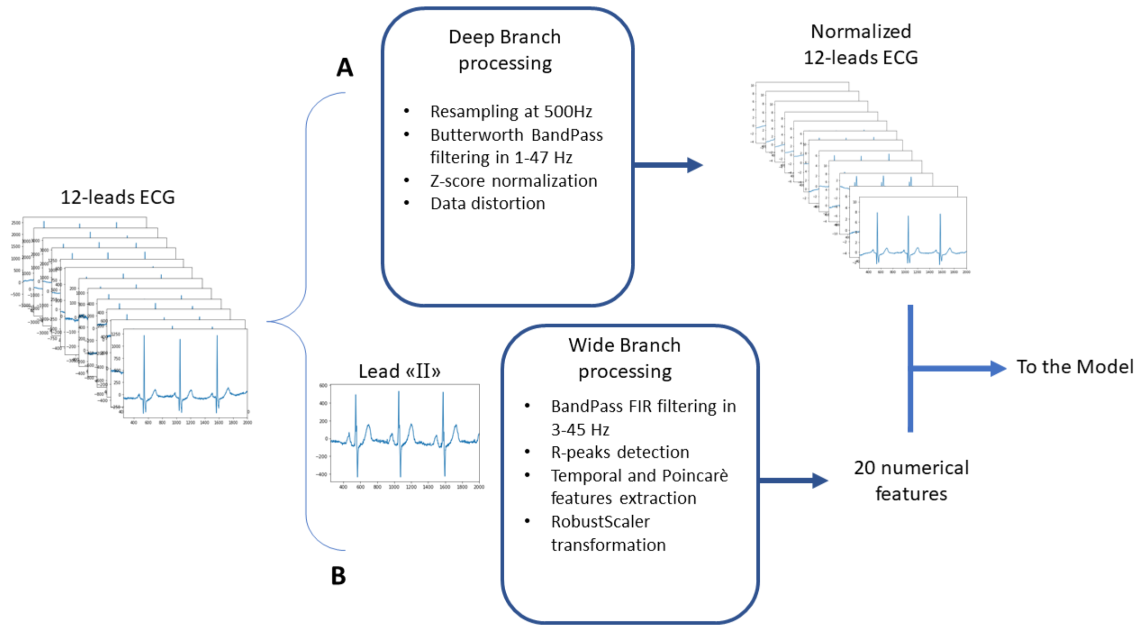


Figure 2.3: A summary of the ECG processing steps operated. Flow A is the path taken by the recording before feeding it to the Deep branch, while flow B is the one followed for the parameters extraction given to the Wide branch.

In particular, the pathway A describes the data preparation steps before feeding the signals to the Deep branch, whose purpose was to recognize the morphological characteristics of each class. Along the other pathway B, the information flow was designed to extract the 20 numerical features used as input of the Wide branch. Such features describe the temporal characteristics of the signal taken into account: in an healthy ECG track they have standard values, which are altered when a certain pathology affects a subject. So the network can learn to recognize the values to classify abnormal recordings represented by the classes of the Challenge.

The **Deep branch** data processing executed on each signal was performed as follows:

1. Resampling at 500 Hz;
2. Filtering with 3^{rd} order Butterworth filter in the range $[1 - 47]$ Hz;

3. Z-score standardization of each channel of the ECG, expressed as follows:

$$Z = \frac{x - \mu}{\sigma},$$

   where μ was the mean of each channel and σ was its standard deviation;

4. Either zero-padding for signals shorter than 5000 samples, or random windowing when their length exceeded 10 seconds.

Even though the data selection procedure explained in section 2.1.2 had already considered 10 seconds 500 Hz signals only, these processing steps were needed to guarantee the standardization of signal characteristics for a generic test set. Moreover, just before giving the ECGs as input in the network, a data distortion procedure was applied with three possible outcomes:

- Random noise addition with a probability of 6%;
- Leads exchange with a probability of 2%;
- Signal inversion over 1, 2 or all the channels with a probability of 2%.

This last strategy was executed to increase the model generalization capabilities, by making the classifier deal with noisy signals.

The processing followed for the Deep branch aimed to the standardization and the noise removal in all the signals, so that even if their origin differed, they would be all similar in shape when fed to the neural network, allowing it to correctly learn the implicit morphological features needed for the classification.

The purpose of the processing for the **Wide branch** was to extract handcrafted features which would help in the temporal characterization of inputs. To compute them, the lead "II" was selected: first, it was filtered with a FIR filter in the range $[3 - 45]$ Hz, then the R peaks were extracted for the computation of RR intervals and Poincaré plot for each recording. Finally, the 20 handcrafted features, listed in table 2.3, were extracted.

Parameter	Description
mean_nni	*Mean NN interval*
sdnn	*NN interval std* [1]
sdsd	*Std of adjacent NN-intervals differences*
nni_50	*Number of NN exceeding 50ms*
pnni_50	*Percentual of nni50 over total RR intervals*
nni_20	*Number of NN exceeding 20ms*
pnni 20	*Percentual of nni20 over total RR intervals*
rmssd	*RMS* [2] *of NN differences*
median_nni	*Median NN interval*
range_nni	*Max-Min difference*
cvsd	*CV* [3] *of successive differences*
cvnni	*Coefficient of Variation*
mean_hr	*Mean Heart Rate*
max_hr	*Max Heart Rate*
min_hr	*Min Heart Rate*
std_hr	*std of Heart Rate*
sd1	*Std of the major axis of Poincaré plot*
sd2	*Std of the minor axis of Poincaré plot*
sd_ratio	*Ratio between sd2 and sd1*
ellipse_area	*Area of fitted ellipse of Poincaré plot*

1. Standard Deviation, 2. Root Mean Square, 3. Coefficient of Variation.

Table 2.3: List of the 20 handcrafed features fed to the Wide branch.

2.3. Network Architecture

As mentioned before, the structure of the Neural Network exploited for the classification of an ECG was composed by two branches, a Deep ResNet SE with dilated convolutions and a Wide Fully Connected Neural Network consisting in 3 sequential layers. The first part received in input the normalized 12-leads signals, while the second one was fed with the 20 numerical features extracted (respectively, flows A and B of figure 2.3). The complete model architecture is represented in figure 2.4, and the code is reported in Appendix A.4.

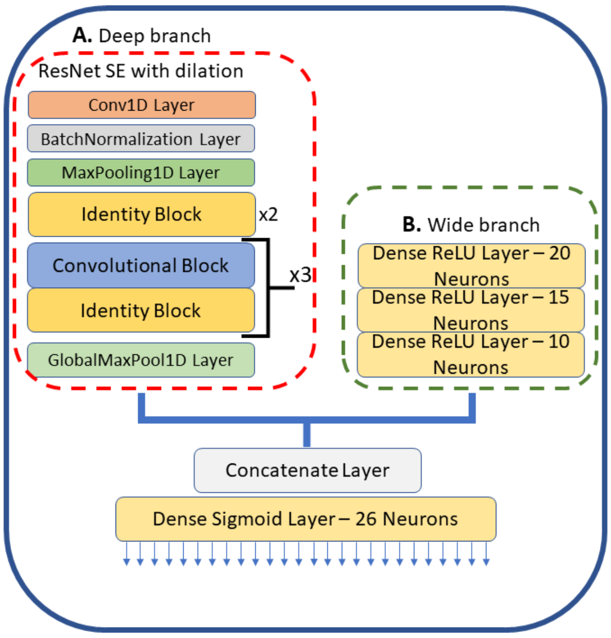


Figure 2.4: Complete model architecture: block A on the left is the Deep branch, made up by the modified ResNet SE receiving the normalized 12-leads ECG; block B on the right is the Wide branch, constituted by the Fully Connected Neural Network that took as input the numerical features and created an embedding of them. Their outputs are concatenated and fed to the final sigmoid layer to perform the prediction.

2.3.1. Deep branch

The **Deep branch** of the model took its foundations on the modified ResNet with Squeeze-and-Excitation (SE) attention mechanism implemented by Zhao et al. in the 2020 PhysioNet Challenge [50]. Its main use was to extract implicit morphological features both in the single channel of an ECG and, thanks to the attention mechanism, also among the various leads.

The name ResNet stays for Residual Network, because these types of NN are composed by blocks, called Residual blocks, whose output is fast-forwarded to deeper layers [19]. By using this type of structures it is possible to create very deep networks, as their optimization is very simple from a computational point of view. In this particular Network, the two Residual Blocks used are called Identity block and Convolutional block; their structure comprised the Squeeze-and-Excitation (SE) block, which implemented an attention mechanism to help the network to focus on relationships among the different channels of the input.
The **Identity block** was made by two Convolutional layers separated by a Batch Nor-

malization layer and a Dropout layer. These last two layers have a regularization purpose to avoid overfitting. Then an SE block was placed just before the final Additional layer, which summed the original input of the Identity block to the output returned by the SE block.
The **Convolutional block** had the same structure of the Identity block, but a further Convolutional layer was placed after the SE block.
Finally, the **SE block** implemented an attention mechanism through a Global Average Pooling layer, which performs a *squeezing* of the received features, and added nonlinearity through two Dense layers with ReLU and Sigmoid activation functions.
The structure of these blocks is represented in figure 2.5. A further improvement carried out over the blocks was the execution of dilated convolutions inside the Identity blocks' convolutional layers: this adjustment allowed to expand the distance considered by the kernels in performing the convolutions, which gave the possibility to watch the signal at different time scales. This way, the ResNet would extract meaningful features in a channel considering windows of different size.

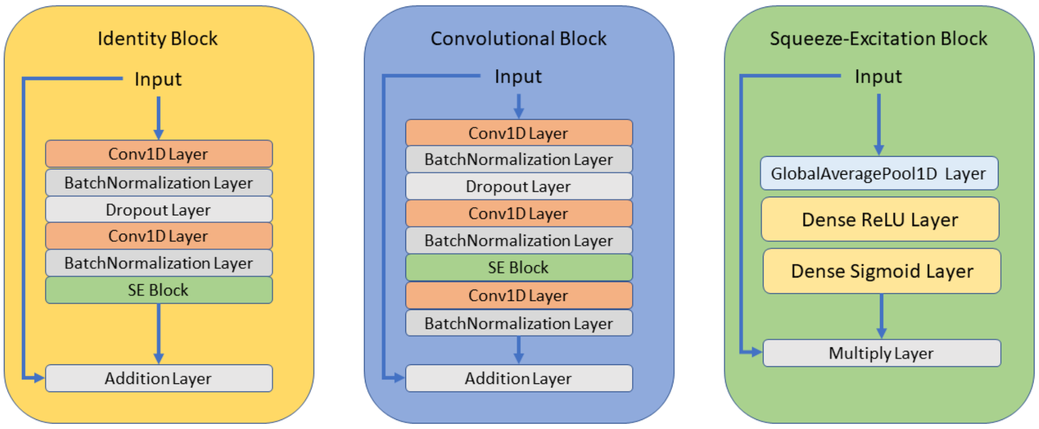


Figure 2.5: Residual blocks architecture: in all the blocks it can be seen how the received input is propagated to the last additional or multiply layer to optimize the training process and allow to increase the number of layers.

From figure 2.4 it can be seen that in block A, which represents the Deep branch, the ECG was first given to a Convolutional layer, followed by a Batch Normalization and a Max Pooling layer. Successively, the Network continued its elaboration with two Identity

blocks and an alternance of a Convolutional and an Identity block for three times. Finally, the extracted features were fed to a Global Max Pooling layer and then concatenated to the outputs of the Wide branch.
The first Convolutional layer of the Network presented 64 filters with a kernel size of 15 and a dilation rate of 4. Successively, the 5 following Identity blocks had an increasing dilation rate of their Convolutional layers with the sequence 4, 8, 16, 32 and 64. Also, the convolutional filters of both the Identity and Residual blocks had a kernel size of 7, but their number was increased by going deeper in the network, using 64, 128, 256 and 512 filters each. This allowed to increase the complexity of extracted features.

2.3.2. Wide branch

The **Wide branch** was implemented to receive in input the 20 numerical handcrafted features obtained at the end of the path B of data processing (figure 2.3), and make an automatic selection of the most representative ones. This was done thanks to the 3 sequential Dense layers with ReLU activation functions, composed respectively by 20, 15 and 10 neurons: this bottleneck structured FCNN (see figure 2.6) forced the layers to synthesize the numerical inputs in the most effective way for the classification of the signals.

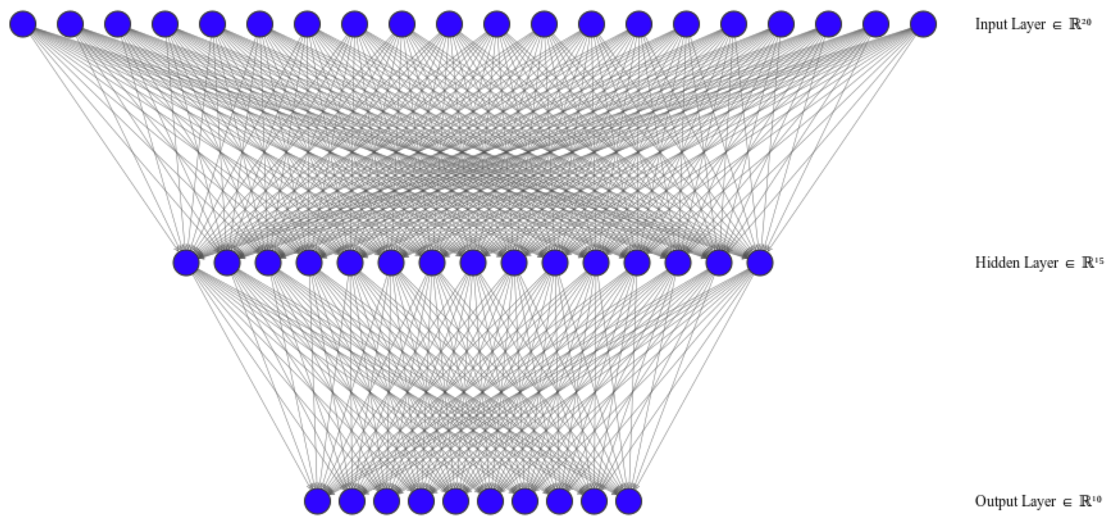


Figure 2.6: Bottleneck architecture of the Wide branch, which forces to learn an embedding of inputs selecting the most significant ones to be returned from the output layer.

As shown in figure 2.4, the resulting outputs were then concatenated to the extracted features of the Deep branch and fed to the final classifier, which consisted in a Dense sigmoid layer made of 26 neurons, one for each class to predict. The sigmoid activation function returned probabilities in the range $[0, 1]$ for each class, complying to the requirement of assigning multiple classes to the same recording.

2.4. Training procedure

Considering the difference in output sizes of the two parts of the model, respectively 512 features for the Deep branch and 10 for the Wide branch, and in the number of trainable parameters, with the ResNet presenting 1000 times the number of parameters of the Wide network, it can be understood how the training could be unbalanced toward the Wide branch. In fact, having much less parameters and being a FCNN, the block B could easily incur in overfitting, preventing the complete training of the branch A. For this reason, a **three-step training** was implemented to allow the two blocks to focus on their training without interfering with the other.

1. First of all, the ResNet was trained for 30 epochs without considering the Wide branch in the model architecture. This allowed to extract the meaningful implicit morphological characteristics of the recordings.
2. Secondly, the previously computed parameters of the Deep branch were loaded in the model comprising also the Wide network and frozen. The Wide branch was trained for 20 epochs: by doing so, the final classifier would consider both the deep morphological features together with the wide numerical ones for the error computation, but only update of the FCNN's parameters.
3. As last step, 10 supplementary epochs were used for the fine tuning of the Deep branch with frozen wide parameters. This ulterior training was done to adapt the deep features to the fully trained wide model.

For the training, a stratified train-validation split was performed, using 20% of the available data as validation set. Early stopping over validation loss was implemented, so that the performances were evaluated on unseen data before choosing to freeze the training. The learning rate was set to 0.003, with a tenfold decay every 10 epochs, but during the fine-tuning of step 3, it was reduced every 2 epochs. Adam optimizer was used, together with binary-crossentropy loss function, particularly indicated for multiclass classification problems. The chosen batch size was 64.

After the model training, a threshold optimization process was performed to improve the

classification performances: having an unbalanced dataset, the final classifier returned higher probabilities to the most numerous classes, predicting with very low values the rarer ones. If the predictive thresholds were left unchanged for all the labels without any consideration, the result would be to neglect the uncommon diseases.
For this reason, the threshold was customized for each sigmoid neuron in the classifier layer so that the predictions would not be affected by the class unbalancing. Two steps were followed:

1. A grid-search of the best threshold vector which maximizes the Challenge Metric (CM) in the range $[0; 4]$ with a step of 0.01. The CM was computed on validation data to keep generality.
2. The application of the Nelder-Mead downhill simplex minimization method over the negative of CM, using the previously computed threshold vector in step 1 as initial vector.

The complete algorithm can be found in the Appendix A.5. These newfound thresholds were useful to obtain more reliable predictions, as well as to maximize the CM score obtained.

2.5. Ensemble evaluation

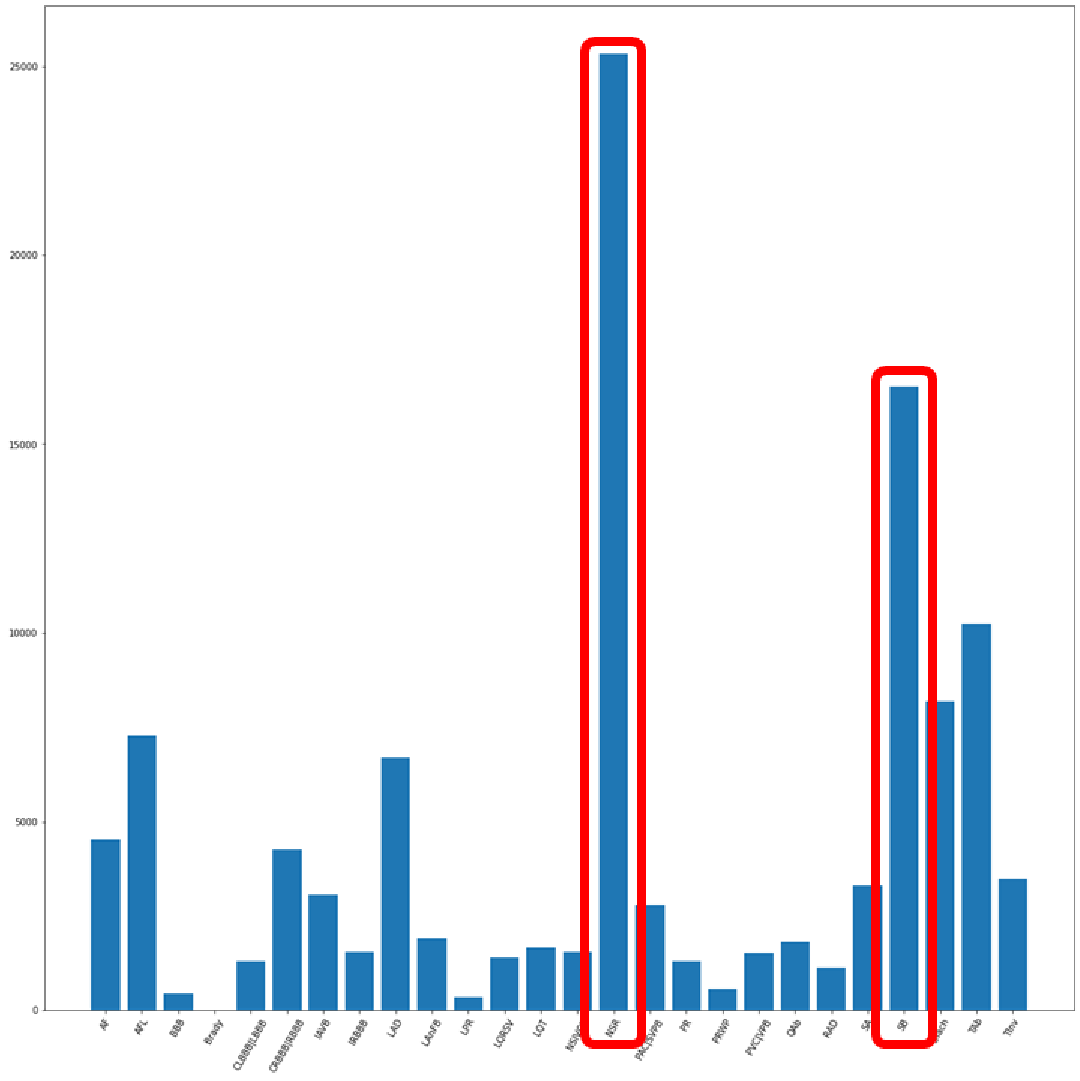


Figure 2.7: Original label distribution in training dataset before reducing NSR and SB.

Figure 2.7 shows the histogram of label distribution in the available training dataset. It is clear how the number of recordings presenting NSR and SB is overwhelming with respect to other classes such as Brady or BBB.

To reduce this unbalancing, the two most represented classes, NSR and SB, were removed from the available data and randomly divided in three subgroups. Thanks to this, three less unbalanced datasets were identified, each containing all the original data, but a different third of the removed NSR and SB, as shown in figure 2.8.

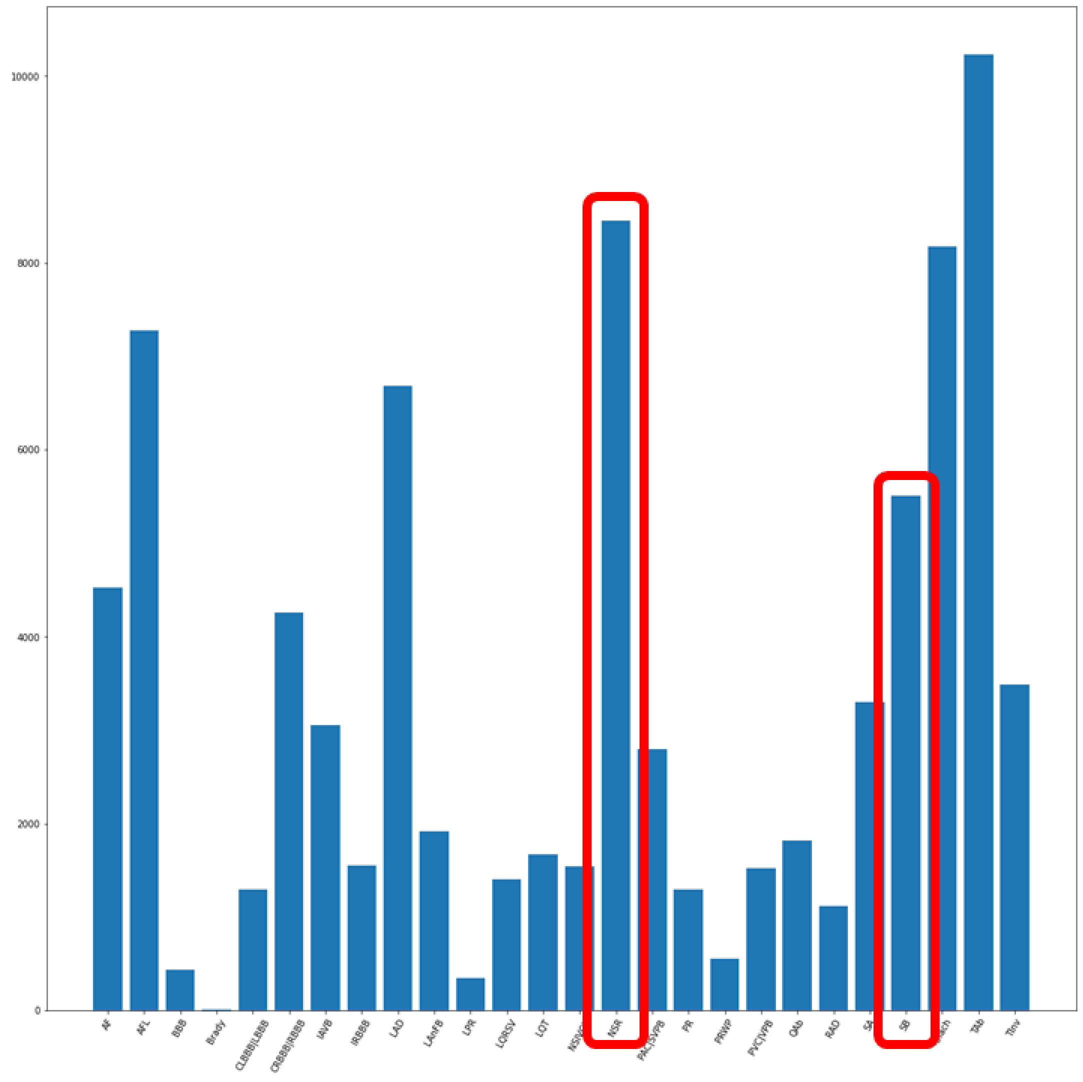


Figure 2.8: Label distribution in training dataset after the reduction of NSR and SB.

Three distinct models were trained using the presented procedure with the new subsets: this way, each network could see an improved distribution of the labels. Moreover, the different thirds of removed labels changed among these models, so during their training they could see different expressions of the same label. By doing so, the three models had a better generalization capability when used together in an ensemble classifier.

To evaluate the ensemble over the test set, the same recording was fed to each model, and the classes were assigned with a majority voting criterion: the class was assigned only if at least two of the three models returned it.

The complete algorithm of this part can be found in the Appendix A.6. Also, figure 2.9 shows a graphical summary of how the ensemble model is used to produce the final prediction over a signal.

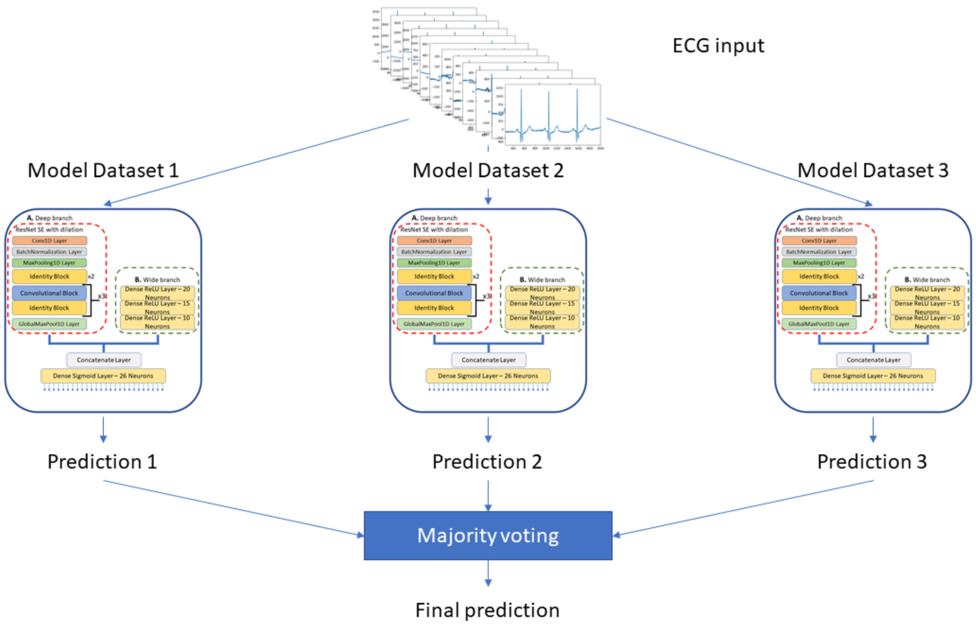


Figure 2.9: The same ECG is given to the three models trained over the different datasets, which will return a prediction each. These predictions will be then used to produce the final classes according to a majority voting criterion.

3 | Results

In this section, the Challenge Metric score will be presented, and the results obtained for the 12-lead and 2-lead models during the Challenge and over the local test set will be reported.

3.1. Challenge Metric

For the PhysioNet Challenge purpose, a new scoring metric which partially rewards mis-diagnoses has been developed: in fact, even if mis-classified, some pathologies can result in similiar treatment or outcome with respect to the true diagnosis. This aspect suggests the fact that mixing up some classes is not as harmful as confusing others, making this score a generalized accuracy with a focus on the medical diagnosis [40]. For this reason, even though it is still an error, confusing Bradycardia (Brady) and Sinus Bradycardia (SB) still rewards the classification of 0.5 points, because it is not as harmful as confusing it with Qwave Abnormality (QAb) (awarded with only 0.2 points).

To compute the actual Challenge Metric a multi-class confusion matrix was produced by counting of the number of predictions per class. In the matrix, the *rows* represented the true labels, while the *columns* were the model outputs. The correct diagnoses are placed on the diagonal of the matrix, while the off-diagonal elements are the mis-classifications. To give different relevance to mis-classifications, different weights were assigned to off-diagonal elements as reported in Figure 3.1.

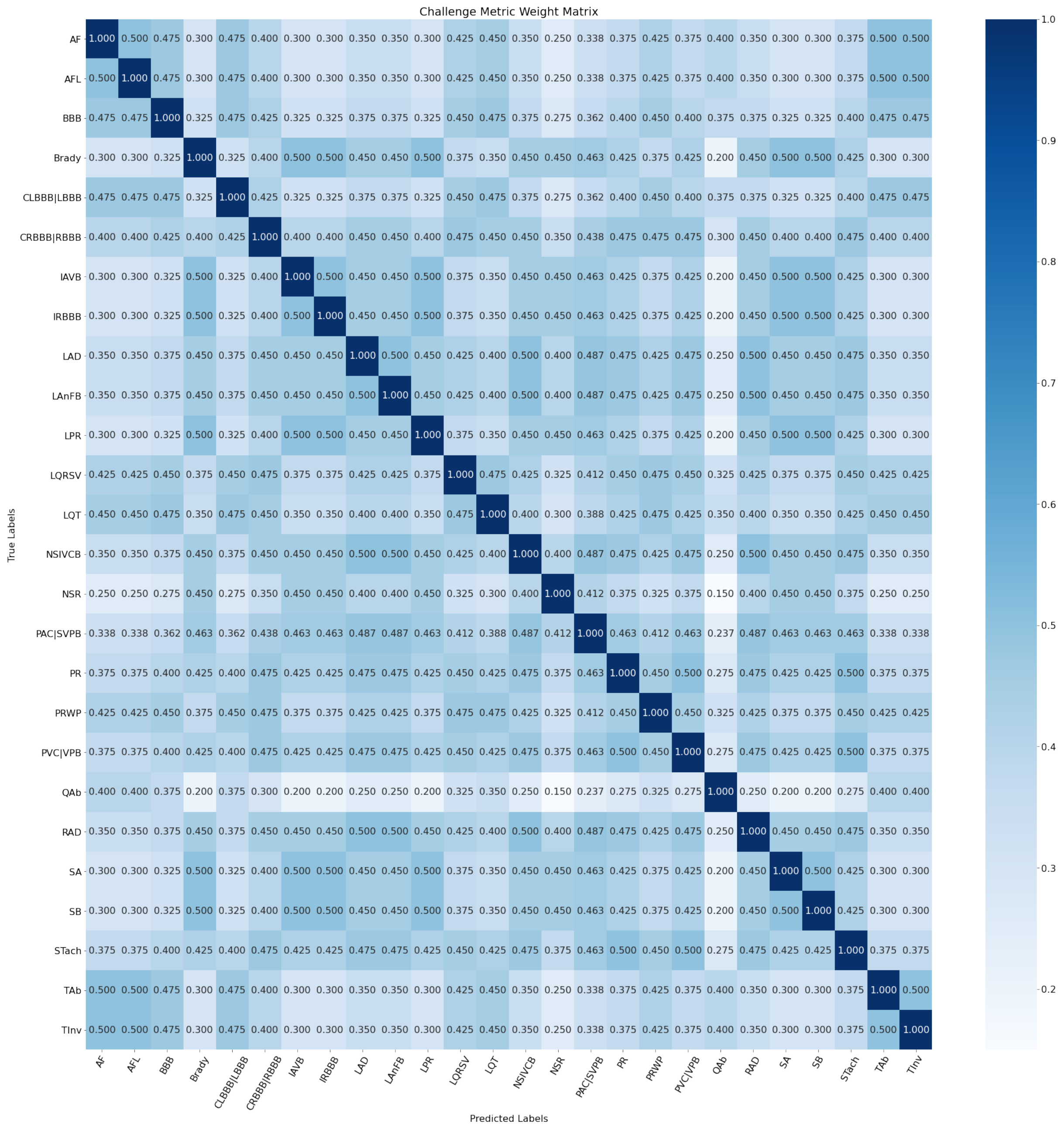


Figure 3.1: This heatmap shows the weights given to each class when they are used for the computation of the Challenge Metric.

The computed score was then evaluated as the sum of the normalized instances of a class multiplied by its weight, according to the following formula:

$$\begin{cases} \boldsymbol{CM} = \sum_{i,j=1}^{K_{classes}} a_{ij} * w_{ij}, & \text{(3.1a)} \\ a_{ij} = c_{ij}/N, & \text{(3.1b)} \\ N = \max\{\sum(\boldsymbol{y} \vee \hat{\boldsymbol{y}}),\ 1\}, & \text{(3.1c)} \end{cases}$$

where $K_{classes}$ is the number of considered classes, in this case 26; a_{ij} is the normalized number of instances of the considered class in all the predictions; w_{ij} is the weight assigned to that class taken by the matrix of Figure 3.1; c_{ij} is the number of instances of that class before normalization; N is the normalization factor defined as the maximum between **1** and the total number of labels assigned to a recording being them either the predicted ($\hat{\boldsymbol{y}}$) or the true ($\boldsymbol{y}$) labels: this was expressed through the "or" ($\vee$) operator between the Boolean vectors $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. The normalization factor N was such that if an algorithm always predicts a wrong class, the score would be decreased, because it would increase the normalization factor penalizing also the true predictions of that recording.
For instance, if the recording was labelled with 3 diagnoses, and the classifier returned those 3 classes and 2 additional wrong labels, then N=5. Thus the confusion matrix element a_{ij} would be divided by 5 and not by 3, thus penalizing also correct classifications.
The Challenge Metric can be thus considered a generalized accuracy for multi-class problems which focuses on the clinical importance of the predictions.

3.2. PhysioNet Challenge evaluation

Recalling the question posed by the PhysioNet/CinC Challenge 2021: "Will Two do? Varying Dimensions in Electrocardiography", the objective of the model evaluation was to understand whether the use of reduced sets of leads for the classification of ECGs produced predictions comparable to those obtained by the complete 12-leads set.
For this reason, models with 12, 6, 4, 3 and 2-leads subsets were trained and evaluated in the test phase. For simplicity, the results presented in this section focused only on 12 and 2 leads, making a direct comparison between their results.

There were three phases to be attended by the participants, during which an evaluation of the algorithms was performed:

1. The **Unofficial phase** had a smaller dataset available for training, and required to train the models for all the previously stated leads subsets, excluding the one for the classification of the 4-leads. The Challenge evaluation was only performed on a hidden validation set. During this phase our team implemented only the ResNet

SE, without using dilated convolutions.

2. The **Official phase**, during which all the available data were given, and all the 5 leads subsets were required. The evaluation was performed on hidden validation and test sets. At our final official submission, the model included both the Deep and Wide branch, again without the dilated convolutions. Also, the training procedure was made of the first two steps, without considering the deep branch finetuning.

3. The **Focus Issue** submission was the last one, where more freedom was given to the participants in terms of requirements. The complete model architecture presented in chapter 2 was developed in this phase and used for the final submission.

The tables with the scores obtained in the different phases are reported in the next sections. They included the evaluation of the models in the three phases on the test sets, for both the 12 and 2-leads subsets. While in the Unofficial and Official phases the models were correctly submitted and the obtained scores are reported in table 3.1 and table 3.2, during the Focus Issue phase there was not an official evaluation for timing problems. For this reason, a local test set was used to obtain the results presented in table 3.3.

To achieve a direct comparison among the models developed during the various phases, the same stratified local test set that was used for the Focus Issue model evaluation was also exploited as test set of the Unofficial and Official phase models, producing further results for their evaluation presented in table 3.4. This additional evaluation was useful to understand whether the changes brought to the Focus Issue model improved the classification capability with respect to the algorithms of the other phases.

3.2.1. Unofficial Phase Results

In table 3.1 are reported the results for the 12 and 2-leads subsets evaluated in the Unofficial phase. The scores were obtained on the Challenge Validation set, and the model used only included the Deep ResNet SE without Wide branch and with no dilated convolutions.

Unofficial Phase submission	
Leads	Challenge Validation set
12	0.579
2	0.555

Table 3.1: In the table is reported the Challenge Metric evaluated over the Challenge Validation set during the Unofficial phase.

These results show that the reduction of the leads brings a slight drop in performances, meaning that some information is lost when moving from 12 to 2 leads. However, the score obtained with the 2-leads model is not too far from the one returned by the 12-leads model. This last consideration gave confidence that the use of the Wide branch could fill the gap in the information lost in the selection of the lead subset.

3.2.2. Official Phase Results

In table 3.2 the results obtained by the *PhysioNauts* team, as well as the 1st classified ISIBrno-AIMT, during the Official Phase are reported. Here our model included the Deep + Wide branch, without the dilated convolutions and with only 16 temporal features. Due to issues with the UMich dataset during testing, our model did not receive an official ranking. However, different Challenge Test sets were evaluated by using the model. The results allowed to see how the model behaved by using different sources of data, thus to see its generalization capabilities. Also, a comparison between the performances of our model and the ones scored by the winners of the challenge on those Test sets can be observed.

Official Phase submission - PhysioNauts				
Leads	Local Cross-validation	Challenge Validation set	Challenge 1st Test set [1.]	Challenge 2nd Test set [2.]
12	0.689 ± 0.004	0.613	0.710	0.590
2	0.656 ± 0.007	0.582	0.630	0.570

1. CPSC Hidden Test set, 2. Georgia 12-leads ECG Hidden Test set.

Official Phase submission - ISIBrno-AIMT (ranked 1st)				
Leads	Local Cross-validation	Challenge Validation set	Challenge 1st Test set [1.]	Challenge 2nd Test set [2.]
12	0.69 [3]	0.640	0.730	0.620
2	NA [3]	0.620	0.690	0.600

1. CPSC Hidden Test set, 2. Georgia 12-leads ECG Hidden Test set, 3. Taken from their preprint [35], only 12-leads available.

Table 3.2: In the table are reported the Challenge scores evaluated with local 5-fold cross-validation, Challenge Validation and Test sets of our team. The second table shows the results obtained by the winners of the challenge, allowing to compare the two models performances.

By comparing the Unofficial and Official results obtained by our model in the Challenge Validation set, it can be seen how the introduction of the Wide branch brings an improvement in the classification capabilities of all the leads subsets. Even so, this improvement shows that the Wide branch was not able to compensate the missing information due to the reduction of the number of leads, because there is still a drop of performances which varies according to the used dataset in the 2-leads models.

Still, a good classification capability was obtained by our team, especially if compared with other submissions in the same test sets. For instance, the first classified team, **ISIBrno-AIMT**, obtained for the 12-leads model a score of 0.64 on the Validation set, 0.73 over the CPSC test set, and 0.62 over the Georgia 12-leads test set. This means that their model was better than the one developed by our team of approximately 0.03 points on the considered test sets, which is not an excessively large gap. Thus, it can be asserted that the Deep + Wide model is almost on the same level of the state-of-the-art capability of pathology recognition, even though there is still space for improvements, some of which have been performed during the Focus Issue submission phase.

3.2.3. Focus Issue Phase Results

To evaluate the final model developed for the Focus Issue phase, the stratified local test set extracted at the beginning of data processing was used. In fact, even though the model was correctly submitted, it did not respect the temporal constrains given to complete the training phase, thus it did not receive scoring on the Challenge test sets. In table 3.3 are reported the results obtained evaluating the complete trained model on local validation and test data.

Focus Issue submission						
Leads	Local Validation set			Local Test set		
	D	D + W	D + W + D	D	D + W	D + W + D
12	0.699 ± 0.004	0.696 ± 0.003	0.697 ± 0.003	0.704	0.701	0.705
2	0.672 ± 0.005	0.670 ± 0.005	0.674 ± 0.004	0.674	0.673	0.674

Table 3.3: In the table the Challenge Metric scores of the Focus Issue phase are reported. They were evaluated considering the three models of the ensemble over the local validation set and local test set. The three steps of training are also included, showing the first step of only deep network results (D), the introduction of the wide branch (D + W) and the finetuning of the ResNet (D + W + D).

From the local validation test, a slight improvement of the model can be seen with respect to the Official phase submission. The main differences between the models of the two phases was an improved data processing, the use of dilated convolutions, an increase in the number of wide features and the use of the ensemble trained on the three customized datasets. All these elements helped in the increase of generalization of the model, especially considering the 2-leads improvements. For a direct comparison of the models developed in the various phases, the same local test set was used to evaluate them. The results obtained are reported in table 3.4.

Local Test set			
Leads	Unofficial Phase model	Official Phase model	Focus Issue model [1]
12	0.520	0.643	0.705
2	0.505	0.633	0.674

1. D + W + D model taken as reference.

Table 3.4: In the table are reported the Challenge Metric results obtained with the evaluation of the three models over the same Local Test set.

It can be seen that from the Unofficial to the Official phase a clear improvement occurred thanks to the introduction of the two-branches structure in the model. Also, the changes brought to the Focus Issue model gave an ulterior boost in the classifier's performances.

Due to the unavailability of an official scoring, other metrics had to be explored to better understand the model behavior. Moreover, the Challenge Metric score does not completely explain the model behavior toward each class, because it is an evaluation of the goodness of the predictions from a clinical point of view. For this reason, the Positive Predictive Values (PPV) of each class have been computed to understand the model performances. The values are reported in table 3.5.
By focusing on some of the pathologies, such as AF or AFL, it can be seen how the use of the Wide branch slightly increases the predictive capabilities of the algorithm. This may be due to the double nature of those pathologies, as reported in table 2.2, better recognized with the use of the temporal features. However, the classification of other CVDs, such as the BBB, is worsened with the introduction of the Wide branch, probably due to an incorrect integration of the Wide and the Deep features. To overcome this limitation was implemented the third step of training, which focuses on the finetuning of the Deep branch considering also the Wide characteristics. It can be noticed that the recognition of BBB is improved in the third phase of training, supporting such hypothesis.

Labels	12-leads			2-leads			N. Rec
	D	D+W	D+W+D	D	D+W	D+W+D	
AF	0.345	0.361	0.369	0.343	0.370	0.380	554
AFL	0.661	0.670	0.668	0.638	0.640	0.624	1040
BBB	0.276	0.240	0.260	0.242	0.169	0.225	62
CLBBB | LBBB	0.630	0.635	0.574	0.536	0.461	0.423	163
CRBBB | RBBB	0.688	0.697	0.695	0.629	0.609	0.630	453
IAVB	0.504	0.539	0.515	0.501	0.469	0.474	374
IRBBB	0.383	0.357	0.376	0.266	0.238	0.263	220
LAD	0.516	0.532	0.529	0.457	0.492	0.486	954
LAnFB	0.393	0.410	0.428	0.343	0.381	0.378	273
LPR	0.164	0.216	0.200	0.144	0.173	0.181	49
LQRSV	0.262	0.252	0.254	0.186	0.212	0.198	199
LQT	0.317	0.344	0.331	0.255	0.222	0.272	237
NSIVCB	0.328	0.326	0.349	0.242	0.208	0.201	220
NSR	0.901	0.907	0.906	0.885	0.873	0.895	3537
PAC | SVPB	0.515	0.556	0.543	0.496	0.488	0.526	337
PR	0.799	0.740	0.807	0.815	0.856	0.778	185
PRWP	0.151	0.158	0.160	0.125	0.115	0.149	80
PVC | VPB	0.530	0.533	0.463	0.454	0.530	0.483	217
QAb	0.350	0.364	0.356	0.333	0.321	0.342	259
RAD	0.407	0.394	0.386	0.382	0.411	0.410	159
SA	0.531	0.512	0.542	0.510	0.551	0.566	472
SB	0.943	0.927	0.936	0.945	0.938	0.940	2358
STach	0.881	0.875	0.865	0.857	0.852	0.865	1167
TAb	0.410	0.405	0.408	0.377	0.369	0.381	1461
TInv	0.296	0.290	0.297	0.260	0.253	0.255	497

Table 3.5: Here are reported the 12 and 2-leads Positive Predictive Values computed over each class, distinguishing the Deep branch only, Deep+Wide branches and Deep+Wide with finetuning of the deep part.

The following figures 3.2, 3.3, 3.4, 3.5, 3.6 and 3.7 show the confusion matrices obtained respectively from the first, second and third step of training for both the 12 and 2-leads subsets. The diagonal represents the true predictions, while the other squares represent mis-classifications. The rows are the true labels, while the columns represent the predictions. For this reason, it can be understood that the perfect classifier will present a confusion matrix in which only the diagonal is visible, while the off-diagonal elements are null.

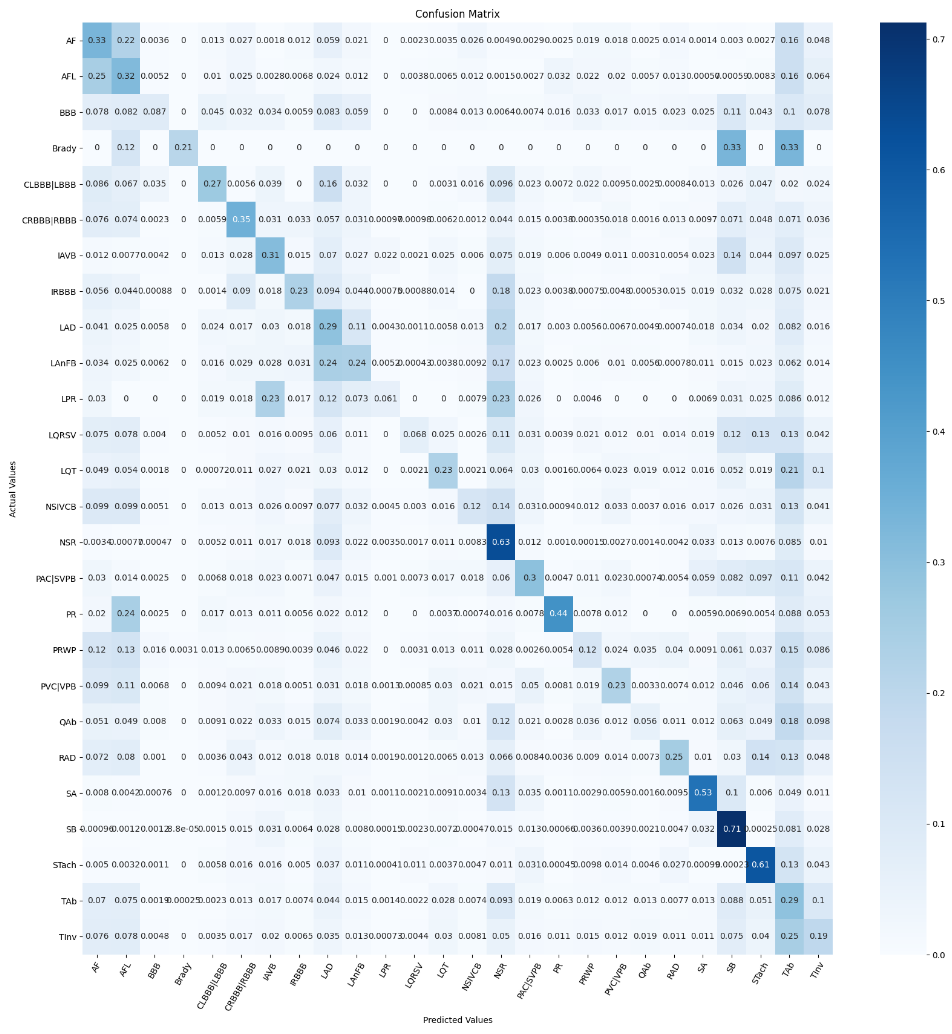


Figure 3.2: The 12-leads Confusion Matrix for the deep model evaluated on the test set.

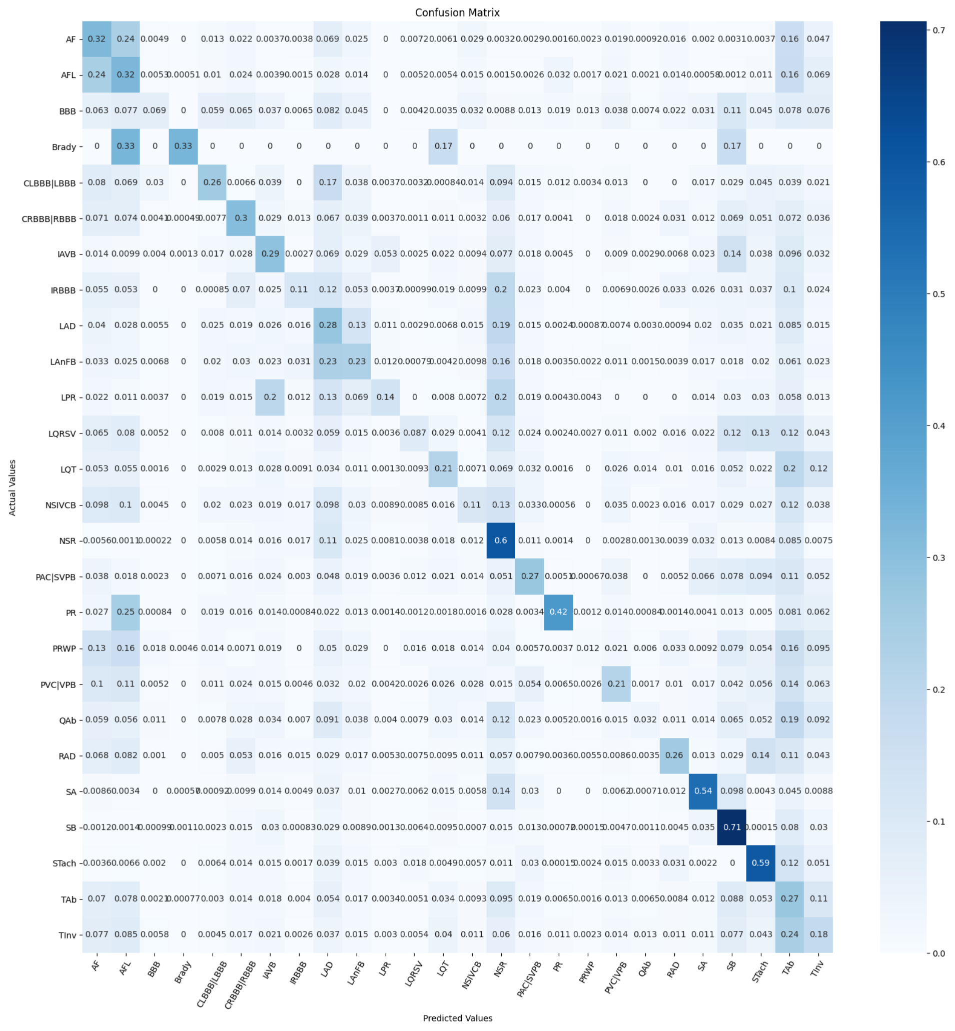
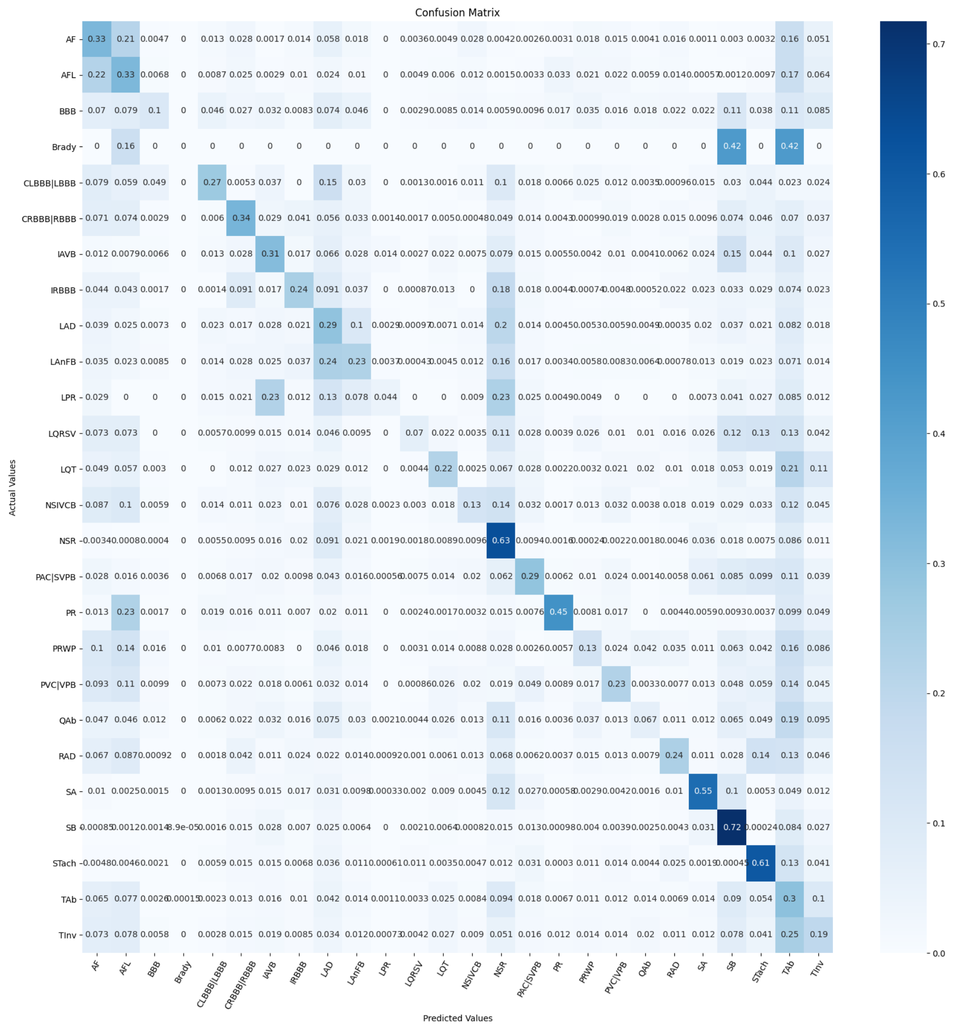


Figure 3.3: The 2-leads Confusion Matrix for the deep model evaluated on the test set.



	AF	AFL	BBB	Brady	CLBBB|LBBB	CRBBB|RBBB	IAVB	IRBBB	LAD	LAnFB	LPR	LQRSV	LQT	NSIVCB	NSR	PAC|SVPB	PR	PRWP	PVC|VPB	QAb	RAD	SA	SB	STach	TAb	TInv
AF	0.33	0.21	0.0047	0	0.013	0.028	0.0017	0.014	0.058	0.018	0	0.0036	0.0049	0.028	0.0042	0.0026	0.0031	0.018	0.015	0.0041	0.016	0.0011	0.003	0.0032	0.16	0.051
AFL	0.22	0.33	0.0068	0	0.0087	0.025	0.0029	0.01	0.024	0.01	0	0.0049	0.006	0.012	0.0015	0.0033	0.033	0.021	0.022	0.0059	0.014	0.00057	0.0012	0.0097	0.17	0.064
BBB	0.07	0.079	0.1	0	0.046	0.027	0.032	0.0083	0.074	0.046	0	0.0029	0.0085	0.014	0.0059	0.0096	0.017	0.035	0.016	0.018	0.022	0.022	0.11	0.038	0.11	0.085
Brady	0	0.16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.42	0	0.42	0
CLBBB|LBBB	0.079	0.059	0.049	0	0.27	0.0053	0.037	0	0.15	0.03	0	0.0013	0.0016	0.011	0.1	0.018	0.0066	0.025	0.012	0.0035	0.00096	0.015	0.03	0.044	0.023	0.024
CRBBB|RBBB	0.071	0.074	0.0029	0	0.006	0.34	0.029	0.041	0.056	0.033	0.0014	0.0017	0.005	0.00048	0.049	0.014	0.0043	0.00099	0.019	0.0028	0.015	0.0096	0.074	0.046	0.07	0.037
IAVB	0.012	0.0079	0.0066	0	0.013	0.028	0.31	0.017	0.066	0.028	0.014	0.0027	0.022	0.0075	0.079	0.015	0.0055	0.0042	0.01	0.0041	0.0062	0.024	0.15	0.044	0.1	0.027
IRBBB	0.044	0.043	0.0017	0	0.0014	0.091	0.017	0.24	0.091	0.037	0	0.00087	0.013	0	0.18	0.018	0.0044	0.00074	0.0048	0.00052	0.022	0.023	0.033	0.029	0.074	0.023
LAD	0.039	0.025	0.0073	0	0.023	0.017	0.028	0.021	0.29	0.1	0.0029	0.00097	0.0071	0.014	0.2	0.014	0.0045	0.0053	0.0059	0.0049	0.00035	0.02	0.037	0.021	0.082	0.018
LAnFB	0.035	0.023	0.0085	0	0.014	0.028	0.025	0.037	0.24	0.23	0.0037	0.00043	0.0045	0.012	0.16	0.017	0.0034	0.0058	0.0083	0.0064	0.00078	0.013	0.019	0.023	0.071	0.014
LPR	0.029	0	0	0	0.015	0.021	0.23	0.012	0.13	0.078	0.044	0	0	0.009	0.23	0.025	0.0049	0.0049	0	0	0	0.0073	0.041	0.027	0.085	0.012
LQRSV	0.073	0.073	0	0	0.0057	0.0099	0.015	0.014	0.046	0.0095	0	0.07	0.022	0.0035	0.11	0.028	0.0039	0.026	0.01	0.01	0.016	0.026	0.12	0.13	0.13	0.042
LQT	0.049	0.057	0.003	0	0	0.012	0.027	0.023	0.029	0.012	0	0.0044	0.22	0.0025	0.067	0.028	0.0022	0.0032	0.021	0.02	0.01	0.018	0.053	0.019	0.21	0.11
NSIVCB	0.087	0.1	0.0059	0	0.014	0.011	0.023	0.01	0.076	0.028	0.0023	0.003	0.018	0.13	0.14	0.032	0.0017	0.013	0.032	0.0038	0.018	0.019	0.029	0.033	0.12	0.045
NSR	0.0034	0.0008	0.0004	0	0.0055	0.0095	0.016	0.02	0.091	0.021	0.0019	0.0018	0.0089	0.0096	0.63	0.0094	0.0016	0.00024	0.0022	0.0018	0.0046	0.036	0.018	0.0075	0.086	0.011
PAC|SVPB	0.028	0.016	0.0036	0	0.0068	0.017	0.02	0.0098	0.043	0.016	0.00056	0.0075	0.014	0.02	0.062	0.29	0.0062	0.01	0.024	0.0014	0.0058	0.061	0.085	0.099	0.11	0.039
PR	0.013	0.23	0.0017	0	0.019	0.016	0.011	0.007	0.02	0.011	0	0.0024	0.0017	0.0032	0.015	0.0076	0.45	0.0081	0.017	0	0.0044	0.0059	0.0093	0.0037	0.099	0.049
PRWP	0.1	0.14	0.016	0	0.01	0.0077	0.0083	0	0.046	0.018	0	0.0031	0.014	0.0088	0.028	0.0026	0.0057	0.13	0.024	0.042	0.035	0.011	0.063	0.042	0.16	0.086
PVC|VPB	0.093	0.11	0.0099	0	0.0073	0.022	0.018	0.0061	0.032	0.014	0	0.00086	0.026	0.02	0.019	0.049	0.0089	0.017	0.23	0.0033	0.0077	0.013	0.048	0.059	0.14	0.045
QAb	0.047	0.046	0.012	0	0.0062	0.022	0.032	0.016	0.075	0.03	0.0021	0.0044	0.026	0.013	0.11	0.016	0.0036	0.037	0.013	0.067	0.011	0.012	0.065	0.049	0.19	0.095
RAD	0.067	0.087	0.00092	0	0.0018	0.042	0.011	0.024	0.022	0.014	0.00092	0.001	0.0061	0.013	0.068	0.0062	0.0037	0.015	0.013	0.0079	0.24	0.011	0.028	0.14	0.13	0.046
SA	0.01	0.0025	0.0015	0	0.0013	0.0095	0.015	0.017	0.031	0.0098	0.00033	0.002	0.009	0.0045	0.12	0.027	0.00058	0.0029	0.0042	0.0016	0.01	0.55	0.1	0.0053	0.049	0.012
SB	0.00085	0.0012	0.0014	8.9e-05	0.0016	0.015	0.028	0.007	0.025	0.0064	0	0.0021	0.0064	0.00082	0.015	0.013	0.00098	0.004	0.0039	0.0025	0.0043	0.031	0.72	0.00024	0.084	0.027
STach	0.0048	0.0046	0.0021	0	0.0059	0.015	0.015	0.0068	0.036	0.011	0.00061	0.011	0.0035	0.0047	0.012	0.031	0.0003	0.011	0.014	0.0044	0.025	0.0019	0.00045	0.61	0.13	0.041
TAb	0.065	0.077	0.0026	0.00015	0.0023	0.013	0.016	0.01	0.042	0.014	0.0011	0.0033	0.025	0.0084	0.094	0.018	0.0067	0.011	0.012	0.014	0.0069	0.014	0.09	0.054	0.3	0.1
TInv	0.073	0.078	0.0058	0	0.0028	0.015	0.019	0.0085	0.034	0.012	0.00073	0.0042	0.027	0.009	0.051	0.016	0.012	0.014	0.014	0.02	0.011	0.012	0.078	0.041	0.25	0.19

Figure 3.4: The 12-leads Confusion Matrix for the wide model evaluated on the test set.

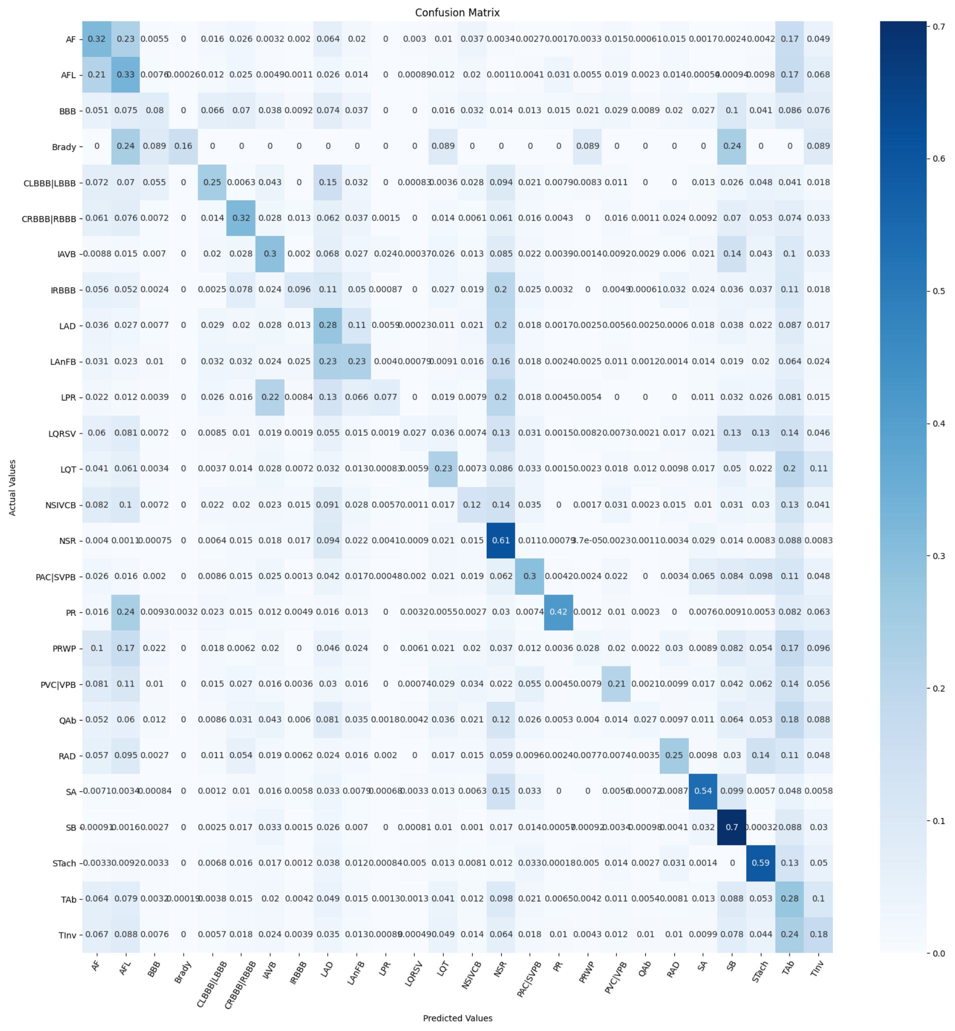


Figure 3.5: The 2-leads Confusion Matrix for the wide model evaluated on the test set.

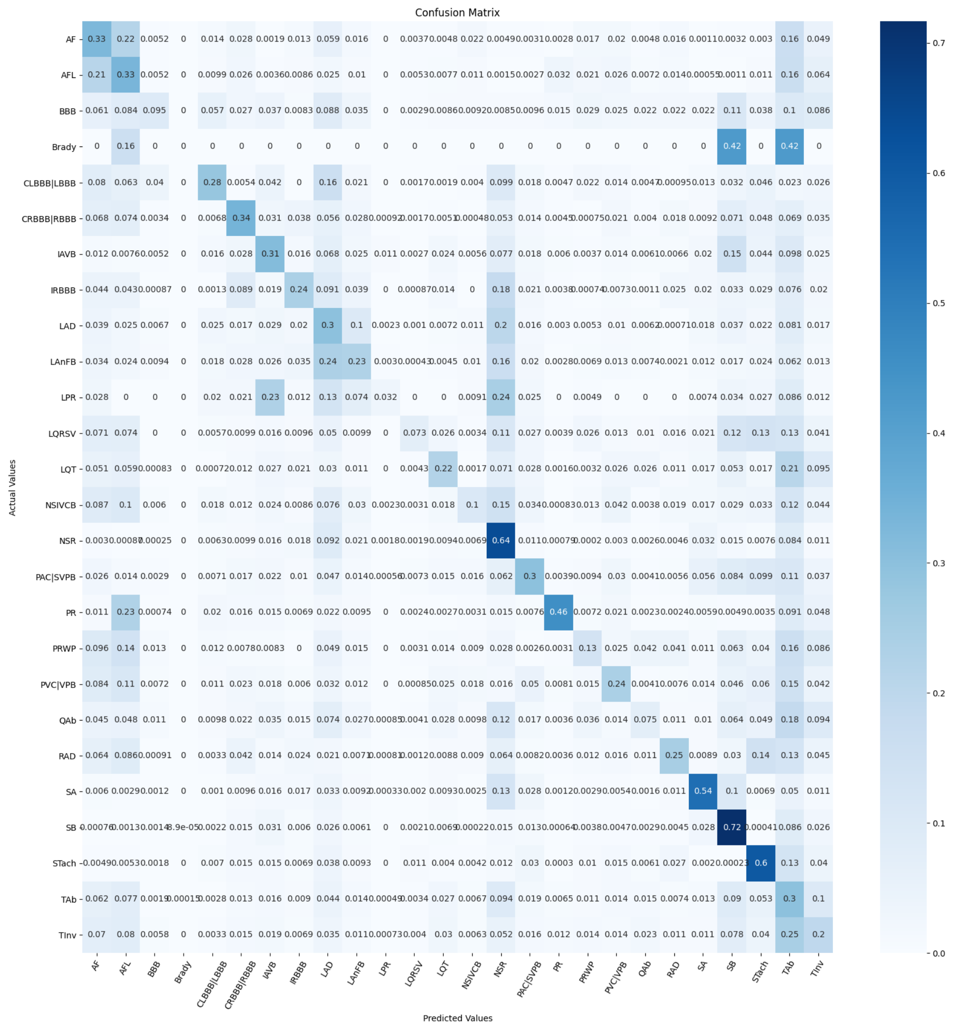


Figure 3.6: The 12-leads Confusion Matrix for the finetuning model evaluated on the test set.

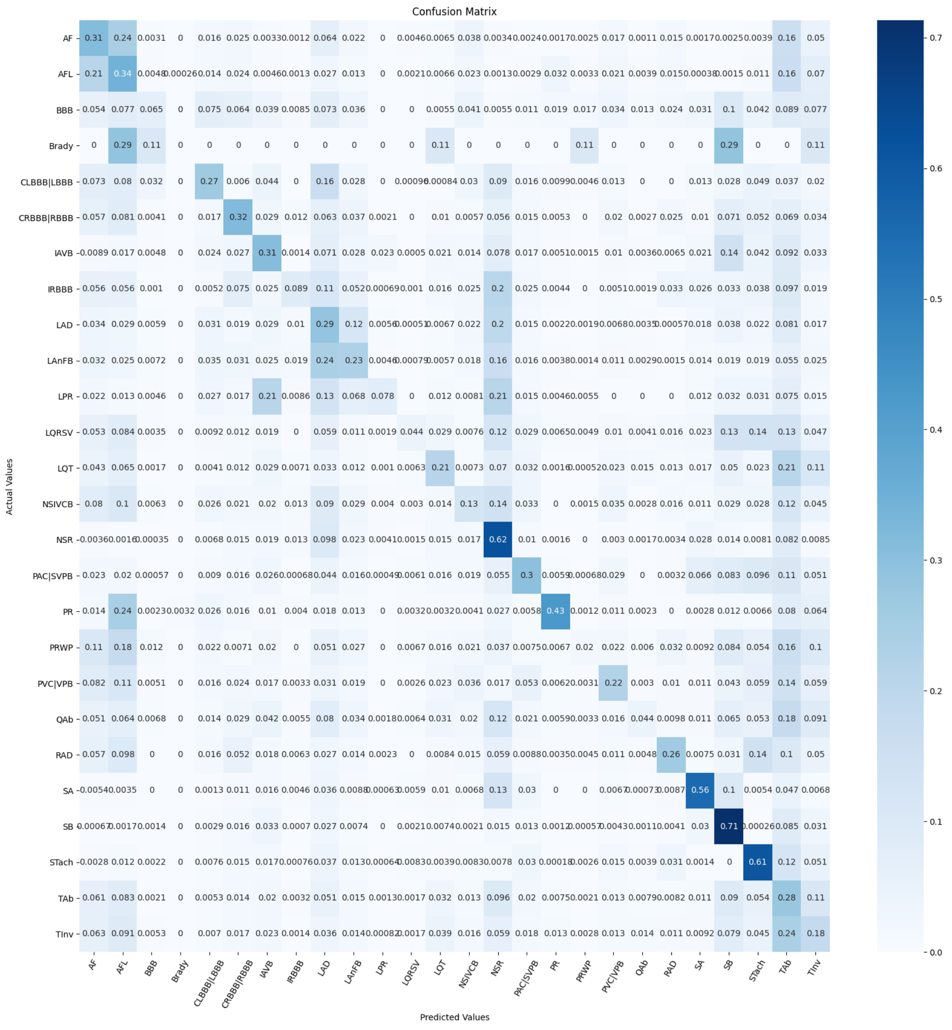


Figure 3.7: The 2-leads Confusion Matrix for the finetuning model evaluated on the test set.

By comparing the confusion matrices of the models used to evaluate the test set after each training phase, it is difficult to notice the differences. Among the matrices it can be seen how most of the predictions are correct, because the diagonals are predominant in the graphs, meaning that the predictions are overall accurate. However, there are some areas in which the mis-classifications are more evident: in the upper left corner of the matrices, and along the first two columns, there is a predominance of light blue squares. This means that the AFL and AF labels are often mixed up, and also assigned when they are not present in the ECG. Other classes, such as LAD, NSR, SB, STach, TAb and TInv are also frequently chosen, producing mis-classifications.

The confusion matrices can be compared, showing the behavior of the models with each class during the various phases of the three-step training. To better understand them in details, the matrices should be considered together with the PPV values: most of the classes benefited from the introduction of the Wide branch, reducing mis-classifications. In particular, for the 12-leads Wide+Deep model the PPV increased for 14 classes with respect to the only Deep branch, while it was reduced in the other cases. Out of those 14 classes, 7 of them were considered as pathologies affecting the heart *rhythm*, supporting the hypothesis that this part of the network can increase the predictive capabilities and increase generalization of the model, especially toward these diseases. Moreover, in 10 out of the 12 remaining classes, even though their PPVs were slightly worsened by the inclusion of the Wide branch, this trend was corrected and stabilized by the finetuning performed over the Deep branch during the third phase of training. However, it has to be noticed that in 8 classes the retraining of the Deep branch reduced the predictive capability of the model.

By considering the 2-leads, a similiar behavior can be noticed with respect to the 12-leads model, but the introduction of the Wide model worsened the predictive capability of 15 classes out of the 26. Again, the finetuning of the Deep model helped to correct this trend in most cases, and for 16 pathologies the recognition capability was improved with respect to the only Deep model.
These observations support the supposition that the retrain of the ResNet can be useful after the inclusion of the Wide features in the classification layer to correct the possible unbalancing of the network toward this last branch.

Combining all the observed results, it is possible to conclude that the model ensemble, integrating both the Wide and Deep features, if trained with the three-step procedure can be a good solution for the classification of ECG pathologies, even compared to the considered state-of-the-art model which won the PhysioNet Challenge 2021. However, there is still space for further improvements, especially if such models aim to be considered for clinical screening.

4 | Conclusions and future developments

In this work an algorithm which wanted to face the problem of ECG automatic classification, combining Machine Learning with the novel techniques of Deep Learning, was presented. This was done to understand whether the model diagnoses could be brought to a level comparable to those of an expert cardiologist even in complex cases where the patient is affected by multiple pathologies. Through the combination of ML and DL, an aspect considered was the development of an algorithm whose classification criterion could be understandable by an external user: through the use of ML features, it can be possible to understand the reason why a certain class was assigned to the ECG.
In particular, the developed Network included a modified ResNet SE, whose usefulness in multi-lead ECG classification had been already proved in the past studies, thanks to its capability to extract implicit morphological features along a single channel and among the various leads [50]. Through this study, it was explored the possibility of improving the generality of the classifier through the concatenation of handcrafted temporal features, which are not usually considered in the deep models [49].
Moreover, the model was developed to participate in the PhysioNet Challenge 2021, in which the impact of the reduction of the leads used for the diagnosis was to be explored. This other aspect was addressed to understand how much information is not redundant, but actually useful for the algorithm's classifications.

The final algorithm encompassed a two-branch Network, combining a modified deep ResNet SE with dilated convolutions and a wide 3-layers FCNN. This particular two-sided structure was exploited to tackle the first two objectives: the deep branch was a purely Deep Learning network, where the feature extraction is completely hidden inside the computations performed by the neurons; on the other hand, the wide branch exploits handcrafted features, as in Machine Learning, and used a Deep Learning structure to perform a selection and combination of the features. Thus, a combination of Machine Learning together with Deep Learning was performed, and the presented results showed how this integration successfully improves the classification capabilities of the model. In

addition, the use of explicit temporal features helps the understanding of classification criterion of the Network, giong in the direction of the previously stated objective.
Moreover, the changes in the deep branch allowed to extract the morphological features of ECG at different time scales, while the wide part created an embedding of 20 handcrafted temporal features, which were extracted from the RR intervals of lead "II" of each ECG given in input to the model. The direct combination of morphological and temporal features was done to compensate the lack of rhythmic description given by the ResNet SE, improving the recognition of pathologies which affected the rhythm of an ECG, as shown by the computed PPVs of the model reported in table 3.5.

Another novelty proposed by this work was the three-step training procedure: due to the FCNN nature and the smaller number of layers and parameters of the Wide branch with respect to the Deep part of the network, the introduction of the Wide branch pushed the overall network towards overfitting. For this reason, at first the Deep part was trained alone, allowing it to extract the morphological features. Then, the Wide branch only was trained considering the Deep features, and finally a finetuning of the Deep branch was performed to keep into account the Wide features in the classification together with the Deep ones. The hypothesis behind this approach was the gradual reduction of overfitting, because the division in steps allowed the network to focus on each branch separately, learning their features without interfering with the other part. Also, an increase of generalization (and thus performances) was expected among the various steps. Regarding this last aspect, only a slight improvement in the predictive capability of the model in the various steps of training was reached, with some differences between the 12 and 2-leads models. However, it can be said that the 3-step training was overall successful in improving the performances of the algorithm, allowing to correctly integrate deep and handcrafted features in a balanced way, especially if compared to the models of the Official Phase and the Unofficial Phase (see tab 3.4).

A further aspect to take into account was the handling of the unbalanced dataset which was originated by merging recordings coming from different acquisition centers. This problem was faced in three ways: with data selection, which allowed to create an ensemble of models; with data processing, to standardize the differences due to the various sources; and with a threshold customization process for the final classification, allowing to change the predictive threshold according to the considered class, which could be more or less frequent. All the three aspects allowed to have good performances in the test set (see table 3.3): in particular, the use of the ensemble of three models increased the generality of the model, not allowing it to overfit toward the training data only.

A limitation in the procedure, due to the reduced computational power available in the training of the models, was the reduced length of the signals considered in the model: in fact, 10 seconds signals were used, but such windows may be not long enough for the extraction of some other temporal features regarding the ECG. However, this was a good compromise between training time and model performances. Moreover, most of the signals in the dataset had such length, thus the majority of data were included thanks to this criterion. Future developments could take into account even the longer signals, so to expand the horizon of handcrafted characteristics to be taken into account by the model. Also, another flaw directly regards the dataset: even though the threshold optimization process helps in the mitigation of class unbalancing, the availability of data of the less represented pathologies still affects the quality of predictions. Thus, another aim for future works should be the increase of data presenting the less represented labels.

Considering the objective of leads reduction, the Wide+Deep feature integration was expected to support and integrate the missing information due to the exclusion of some channels in the recordings. However, as shown in the PPV table 3.5, the temporal features worsened the classification capability of the 2-leads model in 15 labels out of the 26 considered. This could mainly due to overfitting, but the phenomenon should be explored in future works, because it may help to identify the information's usefulness for the pathology characterization.
Only after retraining the Deep branch the performances were improved, showing performances for the 2-leads model which are not too far from the complete 12-leads. This last observation brought to the conclusion that the reduction of leads is possible, and 2-leads recording may suffice for the classification: in fact the reduced leads algorithm behavior is similar to a model trained over the complete recording. However, the classifier needs to improve its pathology recognition power.

Future works should aim to improve the overall classification performances, keeping the explicit features to make the labelling criterion understandable for an human user. For instance, the number of explicit handcrafted features fed to the wide branch could be increased, and a different structure of this part of the network could be developed to better integrate Wide and Deep blocks. Another possible idea could be to perform a grouping of the pathologies whose labels were better recognized thanks to the introduction of the Wide branch, to train two distinct models and create an ensemble of classifiers which exploits different architectures made by a single Deep network and another one composed by a Deep + Wide network to predict different classes.

Bibliography

[1] Pubmed. URL https://pubmed.ncbi.nlm.nih.gov/.

[2] Cardiovascular diseases (CVDs). URL https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[3] A. L. Aro, O. Anttonen, T. Kerola, M. J. Junttila, J. T. Tikkanen, H. A. Rissanen, A. Reunanen, and H. V. Huikuri. Prognostic significance of prolonged PR interval in the general population. European Heart Journal, 35(2):123–129, May 2013. doi: 10.1093/eurheartj/eht176. URL https://doi.org/10.1093/eurheartj/eht176.

[4] J. Aspuru, A. Ochoa-Brust, R. Félix, W. Mata-López, L. Mena, R. Ostos, and R. Martínez-Peláez. Segmentation of the ECG signal by means of a linear regression algorithm. Sensors, 19(4):775, Feb. 2019. doi: 10.3390/s19040775. URL https://doi.org/10.3390/s19040775.

[5] A. R. Barnes, H. E. Pardee, P. D. White, F. N. Wilson, C. C. Wolferth, and Committee of the American Heart Association for the Standardization of Precordial Leads. Standardization of precordial leads: Supplementary report. American Heart Journal, 15(2):235–239, 1938. ISSN 0002-8703. doi: https://doi.org/10.1016/S0002-8703(38)90860-9. URL https://www.sciencedirect.com/science/article/pii/S0002870338908609.

[6] I. Basheer and M. Hajmeer. Artificial Neural Networks: fundamentals, computing, design, and application. Journal of Microbiological Methods, 43(1):3–31, 2000. ISSN 0167-7012. doi: https://doi.org/10.1016/S0167-7012(00)00201-3. URL https://www.sciencedirect.com/science/article/pii/S0167701200002013. Neural Computting in Micrbiology.

[7] M. Boden. A guide to recurrent neural networks and backpropagation. the Dallas project, 2002.

[8] A. Bollmann, K. Sonne, H.-D. Esperer, I. Toepffer, J. J. Langberg, and H. U. Klein. Non-invasive assessment of fibrillatory activity in patients with paroxysmal and persistent atrial fibrillation using the Holter ECG. Cardiovascular Research, 44

(1):60–66, 10 1999. ISSN 0008-6363. doi: 10.1016/S0008-6363(99)00156-X. URL `https://doi.org/10.1016/S0008-6363(99)00156-X`.

[9] R. Bousseljot, D. Kreiseler, and A. Schnabel. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. Biomedizinische Technik, 40(S1):317–318, 1995.

[10] N. K. Cox. The QT interval. Nursing Made Incredibly Easy!, 9(2):17–21, Mar. 2011. doi: 10.1097/01.nme.0000394049.90368.13. URL `https://doi.org/10.1097/01.nme.0000394049.90368.13`.

[11] Douedi S, Douedi H. P wave. StatPearls Publishing, 2022. URL `https://www.ncbi.nlm.nih.gov/books/NBK551635/`. Accessed: 2022-01-26.

[12] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi. A review on Deep Learning methods for ECG arrhythmia classification. Expert Systems with Applications: X, 7:100033, 2020. ISSN 2590-1885. doi: https://doi.org/10.1016/j.eswax.2020.100033. URL `https://www.sciencedirect.com/science/article/pii/S2590188520300123`.

[13] W. Einthoven, G. Fahr, and A. de Waart. On the direction and manifest size of the variations of potential in the human heart and on the influence of the position of the heart on the form of the electrocardiogram. American Heart Journal, 40(2):163–211, 1950. ISSN 0002-8703. doi: https://doi.org/10.1016/0002-8703(50)90165-7. URL `https://www.sciencedirect.com/science/article/pii/0002870350901657`.

[14] E. Goldberger. The avl, avr, and avf leads: A simplification of standard lead electrocardiography. American Heart Journal, 24(3):378–396, 1942. ISSN 0002-8703. doi: https://doi.org/10.1016/S0002-8703(42)90821-4. URL `https://www.sciencedirect.com/science/article/pii/S0002870342908214`.

[15] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen. Recent advances in convolutional neural networks. Pattern Recognition, 77:354–377, May 2018. ISSN 0031-3203. doi: 10.1016/j.patcog.2017.10.013.

[16] E. Guirguis. Holter Monitoring. Canadian Family Physician, 33:985–992, Apr. 1987. ISSN 0008-350X. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2218473/`.

[17] J. Habetha. The MyHeart Project - Fighting Cardiovascular Diseases by prevention and early diagnosis. In 2006 International Conference of the IEEE Engineering in

Medicine and Biology Society, volume Supplement, pages 6746–6749, 2006. doi: 10.1109/IEMBS.2006.260937.

[18] Y. Hafeez. Sinus bradycardia, Aug 2021. URL https://www.ncbi.nlm.nih.gov/books/NBK493201/.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[20] A. Henning. Sinus tachycardia, Aug 2021. URL https://www.ncbi.nlm.nih.gov/books/NBK553128/.

[21] S. Herzog, C. Tetzlaff, and F. Wörgötter. Evolving artificial neural networks with feedback. Neural Networks, 123:153–162, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.12.004. URL https://www.sciencedirect.com/science/article/pii/S089360801930396X.

[22] S. Jayaraman, U. Gandhi, V. Sangareddi, U. Mangalanathan, and R. M. Shanmugam. Unmasking of atrial repolarization waves using a simple modified limb lead system. The Anatolian Journal of Cardiology, 15(8):605–610, Aug. 2015. doi: 10.5152/akd.2014.5695. URL https://doi.org/10.5152/akd.2014.5695.

[23] T. J. Jun, H. M. Nguyen, D. Kang, D. Kim, D. Kim, and Y.-H. Kim. ECG arrhythmia classification using a 2-D convolutional neural network. arXiv:1804.06812 [cs], Apr. 2018. URL http://arxiv.org/abs/1804.06812. arXiv: 1804.06812.

[24] A. H. Kashou, H. Basit, and L. Chhabra. Electrical right and left axis deviation, Nov 2021. URL https://pubmed.ncbi.nlm.nih.gov/29262101/.

[25] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, may 2015. doi: 10.1038/nature14539.

[26] G. W. Lindsay. Attention in psychology, neuroscience, and machine learning. Frontiers in Computational Neuroscience, 14, 2020. ISSN 1662-5188. doi: 10.3389/fncom.2020.00029. URL https://www.frontiersin.org/article/10.3389/fncom.2020.00029.

[27] F. Liu, C. Liu, L. Zhao, X. Zhang, X. Wu, X. Xu, Y. Liu, C. Ma, S. Wei, Z. He, J. Li, and E. N. Y. Kwee. An Open Access Database for Evaluating the Algorithms of Electrocardiogram Rhythm and Morphology Abnormality Detection. Journal of Medical Imaging and Health Informatics, 8(7):1368—1373, 2018.

[28] R. N. MacAlpin. Significance of abnormal q waves in the electrocardiograms of adults

less than 40 years old. Annals of Noninvasive Electrocardiology, 11(3):203–210, July 2006. doi: 10.1111/j.1542-474x.2006.00105.x. URL `https://doi.org/10.1111/j.1542-474x.2006.00105.x`.

[29] S. B. Maind, P. Wankar, et al. Research paper on basic of artificial neural network. International Journal on Recent and Innovation Trends in Computing and Communication, 2(1):96–100, 2014.

[30] P. Marius, V. Balas, L. Perescu-Popescu, and N. Mastorakis. Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8, 07 2009.

[31] J. H. McAnulty and S. H. Rahimtoola. Bundle branch block. Progress in Cardiovascular Diseases, 26(4):333–354, 1984. ISSN 0033-0620. doi: https://doi.org/10.1016/0033-0620(84)90009-4. URL `https://www.sciencedirect.com/science/article/pii/0033062084900094`.

[32] S. Meek. ABC of clinical electrocardiography: Introduction. i—leads, rate, rhythm, and cardiac axis. BMJ, 324(7334):415–418, Feb. 2002. doi: 10.1136/bmj.324.7334.415. URL `https://doi.org/10.1136/bmj.324.7334.415`.

[33] F. M. Muscato. MULTI LABEL CLASSIFICATION OF 12-LEAD ECG: A DEEP LEARNING APPROACH. Politecnico di Milano, 2020.

[34] A. Natarajan, Y. Chang, S. Mariani, A. Rahman, G. Boverman, S. Vij, and J. Rubin. A wide and deep transformer neural network for 12-lead ecg classification. 12 2020. doi: 10.22489/CinC.2020.107.

[35] P. Nejedly, A. Ivora, R. Smisek, I. Viscor, Z. Koscova, P. Jurak, and F. Plesinger. Classification of ecg using ensemble of residual cnns with attention mechanism. In 2021 Computing in Cardiology (CinC), volume 48, pages 1–4. IEEE, 2021.

[36] Z. Nesheiwat. Atrial fibrillation, Nov 2021. URL `https://www.ncbi.nlm.nih.gov/books/NBK526072/#article-17962.s11`.

[37] S. Osowski, T. Markiewicz, and L. T. Hoai. Recognition and classification system of arrhythmia using ensemble of neural networks. Measurement, 41(6):610–617, 2008. ISSN 0263-2241. doi: https://doi.org/10.1016/j.measurement.2007.07.006. URL `https://www.sciencedirect.com/science/article/pii/S026322410700070X`.

[38] J. Park, J. An, J. Kim, S. Jung, Y. Gil, Y. Jang, K. Lee, and I. young Oh. Study on the use of standard 12-lead ecg data for rhythm-type ecg classification problems. Computer Methods and Programs in Biomedicine, 214:106521,

2022. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2021.106521. URL `https://www.sciencedirect.com/science/article/pii/S0169260721005952`.

[39] E. A. Perez Alday, A. Gu, A. J Shah, C. Robichaux, A.-K. Ian Wong, C. Liu, F. Liu, A. Bahrami Rad, A. Elola, S. Seyedi, Q. Li, A. Sharma, G. D. Clifford, and M. A. Reyna. Classification of 12-lead ECGs: the PhysioNet/Computing in Cardiology Challenge 2020. Physiological Measurement, 41(12):124003, Jan. 2021. ISSN 1361-6579. doi: 10.1088/1361-6579/abc960. URL `https://iopscience.iop.org/article/10.1088/1361-6579/abc960`.

[40] Reyna, Matthew, Sadr, Nadi, Gu, Annie, Perez Alday, Erick Andres, Liu, Chengyu, Seyedi, Salman, Shah, Amit, and Clifford, Gari. Will Two Do? Varying Dimensions in Electrocardiography: the PhysioNet - Computing in Cardiology Challenge 2021, 2021. URL `https://physionet.org/content/challenge-2021/1.02/`. Type: dataset.

[41] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65 6:386–408, 1958.

[42] Y. Sattar and L. Chhabra. Electrocardiogram. In StatPearls. StatPearls Publishing, Treasure Island (FL), 2021. URL `http://www.ncbi.nlm.nih.gov/books/NBK549803/`.

[43] M. Sazli. A brief review of feed-forward neural networks. Communications, Faculty Of Science, University of Ankara, 50:11–17, 01 2006. doi: 10.1501/0003168.

[44] P. P. Shinde and S. Shah. A review of machine learning and deep learning applications. In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pages 1–6, 2018. doi: 10.1109/ICCUBEA.2018.8697857.

[45] T. Szandała. Bio-inspired neurocomputing. Studies in Computational Intelligence, 2021. ISSN 1860-9503. doi: 10.1007/978-981-15-5495-7. URL `https://doi.org/10.1007/978-981-15-5495-7_11`.

[46] V. Tihonenko, A. Khaustov, S. Ivanov, A. Rivin, and E. Yakushenko. St Petersburg INCART 12-lead Arrhythmia Database. PhysioBank, PhysioToolkit, and PhysioNet, 2008. doi: `10.13026/C2V88N`.

[47] P. Wagner, N. Strodthoff, R.-D. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter. PTB-XL, a Large Publicly Available Electrocardiography Dataset. Scientific Data, 7(1):1–15, 2020.

[48] E. S. Winokur, M. K. Delano, and C. G. Sodini. A wearable cardiac monitor for long-term data acquisition and analysis. IEEE Transactions on Biomedical Engineering, 60(1):189–192, 2013. doi: 10.1109/TBME.2012.2217958.

[49] G. Yan, S. Liang, Y. Zhang, and F. Liu. Fusing Transformer Model with Temporal Features for ECG Heartbeat Classification. In 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pages 898–905, 2019. doi: 10.1109/BIBM47256.2019.8983326.

[50] Z. Zhao, H. Fang, S. D. Relton, R. Yan, Y. Liu, Z. Li, J. Qin, and D. C. Wong. Adaptive lead weighted resnet trained with different duration signals for classifying 12-lead ecgs. In 2020 Computing in Cardiology, pages 1–4, 2020. doi: 10.22489/CinC.2020.112.

[51] J. Zheng, H. Cui, D. Struppa, J. Zhang, S. M. Yacoub, H. El-Askary, A. Chang, L. Ehwerhemuepha, I. Abudayyeh, A. Barrett, G. Fu, H. Yao, D. Li, H. Guo, and C. Rakovski. Optimal Multi-Stage Arrhythmia Classification Approach. Scientific Data, 10(2898):1–17, 2020.

[52] J. Zheng, J. Zhang, S. Danioko, H. Yao, H. Guo, and C. Rakovski. A 12-lead Electrocardiogram Database for Arrhythmia Research Covering More Than 10,000 Patients. Scientific Data, 7(48):1–8, 2020.

[53] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization, 2015.

A | Appendix A - Model code

This appendix shows some extracts of code used for the ECG classification. In particular, the three elements shown represent:

- The **model architecture**, explicitly showing the various blocks contained in the network. In particular, the three submissions codes are shown, highlighting the differences in code step by step: the *unofficial* submission included only the Deep branch, the *official* submission included also the wide branch with some changes in the parameters and the *focus issue* submission has a slightly different structure in the deep branch and wide branch components.
- The **threshold optimization** algorithm is reported to show the steps followed in an important part of the model development.
- The **ensemble evaluation** by majority voting is also shown, obtained by loading and exploiting the predictions of the three models per each lead subset.

Model Architecture

First, the Residual Blocks structures are presented, which are the same through all the phases.

Listing A.1: Model Architecture - Residual Blocks

```python
def ResBs_Conv(block_input, num_filters):
    '''
    Convolutional Residual Block
    Inputs:
        block_input: input tensor to the ResNet block
        num_filters: no. of filters/channels in block_input

    Returns:
        relu2: activated tensor after addition with original input
    '''

    # The ResBs block consists of:

    # 0. Filter Block input and BatchNorm
    block_input = Conv1D(num_filters, kernel_size=7, strides=2,  padding
='same')(block_input)
    block_input = BatchNormalization()(block_input)
    # 1. First Convolutional Layer
    conv1 = Conv1D(filters=num_filters, kernel_size=7, padding='same')(
block_input)
    norm1 = BatchNormalization()(conv1)
    relu1 = Activation('relu')(norm1)

    dropout=Dropout(0.2)(relu1)

    # 2. Second Convolutional Layer
    conv2 = Conv1D(num_filters, kernel_size=7, padding='same')(dropout)
    norm2 = BatchNormalization()(conv2)

    # 3. SE block (fucntion defined above)
    se = se_block(norm2, num_filters=num_filters)

    # 4. Summing Layer (adding a residual connection)
    sum = Add()([block_input, se])

    # 5. Activation Layer
    relu2 = Activation('relu')(sum)

    return relu2


def ResBs_Identity(block_input, num_filters):
    '''
    Identity Residual Block
        Inputs:
            block_input: input tensor to the ResNet block
            num_filters: no. of filters/channels in block_input

        Returns:
            relu2: activated tensor after addition with original input
    '''

    # 1. First Convolutional Layer
    conv1 = Conv1D(filters=num_filters, kernel_size=7, padding='same')(
block_input)
```

```
    norm1 = BatchNormalization()(conv1)
    relu1 = Activation('relu')(norm1)

    dropout = Dropout(0.2)(relu1)

    # 2. Second Convolutional Layer
    conv2 = Conv1D(num_filters, kernel_size=7, padding='same')(dropout)
    norm2 = BatchNormalization()(conv2)

    # 3. SE block
    se = se_block(norm2, num_filters=num_filters)

    # 4. Summing Layer (adding a residual connection)
    sum = Add()([block_input, se])

    # 5. Activation Layer
    relu2 = Activation('relu')(sum)

    return relu2


def se_block(block_input, num_filters, ratio=16):
    '''
    Squeeze-and-Excitation Block
        Inputs:
            block_input: input tensor to the squeeze and excitation block
            num_filters: no. of filters/channels in block_input
            ratio: a hyperparameter that denotes the ratio by which no. of channels will be reduced

        Returns:
            se_scale: scaled tensor after getting multiplied by new channel weights
    '''

    # 1. Global AVG Pool 1D that computes average on channels
    se_pool1 = GlobalAveragePooling1D()(block_input)
    flat = Reshape((1, num_filters))(se_pool1)
    # 2. Fully connected with C//ratio x1 and relu as activation
    se_dense1 = Dense(num_filters // ratio, activation='relu')(flat)
    # 3. Fully connected with sigmoidal activation Cx1
    se_dense2 = Dense(num_filters, activation='sigmoid')(se_dense1)
    # 4. The output of the block is then multiplied with the input
    se_scale = multiply([block_input, se_dense2])

    return se_scale
```

Then, the structure of the model of the Unofficial Phase is reported.

Listing A.2: Model Architecture - Unofficial Phase Structure

```
def resnet_se_modified(N=8, ch=12, win_len=4096, num_cat_vars= 2,
   classes=24):
    '''
    Inputs:
        N: number of residual blocks
        ch: number of channels of the signal
        win_len: length of signal given in input
        num_cat_vars: number of categorical variables (sex and age)
        classes: number of labels to predict

    Returns:
        model: the complete model ready for training
    '''

    # B. ECG window input of shape (batch_size,  WINDOW_LEN, CHANNELS)
    ecg_input = Input(shape=(win_len, ch), name='ecg_signal')
    # B.1 Conv
    ecg_branch = Conv1D(filters=64, kernel_size=15, padding='same')(
   ecg_input)
    # B.2 BatchNorm
    ecg_branch = BatchNormalization()(ecg_branch)
    # B.3 Relu
    ecg_branch = Activation('relu')(ecg_branch)
    # B.4 Max Pool
    ecg_branch = MaxPooling1D(pool_size=2, strides = 2)(ecg_branch)
    # B.5 ResBs (x8)
    # The number of filters starts from 64 and doubles every two blocks
    # Halving the dimension at the third, fifth and seventh ResBs

    # define ResBs_identity blocks (N = 1, N = 2)
    ecg_branch = ResBs_Identity(ecg_branch, 64)
    ecg_branch = ResBs_Identity(ecg_branch, 64)

    filters = 64
    M= int((N -2 )/2)
    for i in range(M):

        filters = filters*2
        # define N-th ResBs block
        ecg_branch = ResBs_Conv(ecg_branch, filters)
        ecg_branch = ResBs_Identity(ecg_branch, filters)

    # Sigmoid activation function on the last layer
    ecg_branch = GlobalMaxPooling1D()(ecg_branch)
    # Flatten
    ecg_branch = Flatten()(ecg_branch)
    # HEAD Classifier
    ecg_branch = Dense(classes, activation='sigmoid', name='
   sigmoid_classifier')(ecg_branch)
    # Finally the model is composed by connecting inputs to outputs:
    model = Model(inputs=[ecg_input],outputs=ecg_branch)

    return model
```

Here is reported the model structure of the Official Phase, to make a direct comparison between the two.

Listing A.3: Model Architecture - Official Phase Structure

```
def resnet_se_modified_wide(N=8, ch=12, win_len=4096, num_wide_features=
    16, classes=26):
    '''
    Inputs:
        N: number of residual blocks
        ch: number of channels of the signal
        win_len: length of signal given in input
        classes: number of labels to predict

    Returns:
        model: the complete model ready for training
    '''

    # A. Wide features go into a Fully Connected structure of 10 neurons
    wide_input = Input(shape= (num_wide_features, ), name = '
   wide_features')
    wide_branch = Dense(10, activation='relu')(wide_input)
    wide_branch = Flatten()(wide_branch)

    # B. ECG window input of shape (batch_size,  WINDOW_LEN, CHANNELS)
    ecg_input = Input(shape=(win_len, ch), name='ecg_signal')
    # B.1 Conv
    ecg_branch = Conv1D(filters=64,kernel_size=15, padding = 'same')(
    ecg_input)
    # B.2 BatchNorm
    ecg_branch = BatchNormalization()(ecg_branch)
    # B.3 Relu
    ecg_branch = Activation('relu')(ecg_branch)
    # B.4 Max Pool
    ecg_branch = MaxPooling1D(pool_size=2, strides = 2)(ecg_branch)
    # B.5 ResBs (x8)
    # Here number of filters starts from 64 and doubles every two blocks
    # Max pooling is of size 2
    # Halving the dimension at the third, fifth and seventh ResBs

    # define ResBs_identity blocks (N = 1, N = 2)
    ecg_branch = ResBs_Identity(ecg_branch, 64)
    ecg_branch = ResBs_Identity(ecg_branch, 64)

    filters = 64
    M= int((N -2 )/2)
    for i in range(M):

        filters = filters*2
        # define N-th ResBs block
        ecg_branch = ResBs_Conv(ecg_branch, filters)
        ecg_branch = ResBs_Identity(ecg_branch, filters)

    # reshape_size=int(np.floor(ch/2)*512)
    # Sigmoid activation function on the last layer
    ecg_branch = GlobalMaxPooling1D(name='gmp_layer')(ecg_branch)
    # Flatten
    ecg_branch = Flatten()(ecg_branch)
    # Concatenate
    shared_path = concatenate([ecg_branch, wide_branch], name='
    concat_layer')
    # HEAD Classifier
    shared_path = Dense(classes, activation='sigmoid', name='
    sigmoid_classifier')(shared_path)
    # Finally the model is composed by connecting inputs to outputs:
    model = Model(inputs=[ecg_input,  wide_input],outputs=shared_path)

    return model
```

Finally, the Focus Issue Phase model is reported.

Listing A.4: Model Architecture - Focus Issue Phase Structure

```
def dilationnet_se_modified_wide(N=8, ch=12, win_len=4096,
    num_wide_features= 16, classes=26):
    '''
    Inputs:
        N: number of residual blocks
        ch: number of channels of the signal
        win_len: length of signal given in input
        classes: number of labels to predict

    Returns:
        model: the complete model ready for training
    '''
    # A. Wide features go into a Fully Connected structure of 20-15-10
    neurons

    wide_input = Input(shape= (num_wide_features, ), name = '
    wide_features')
    wide_branch = Dense(20, activation='relu')(wide_input)
    wide_branch = Dense(15, activation='relu')(wide_branch)
    wide_branch = Dense(10, activation='relu')(wide_branch)

    wide_branch = Flatten()(wide_branch)

    # B. ECG window input of shape (batch_size,  WINDOW_LEN, CHANNELS)
    ecg_input = Input(shape=(win_len, ch), name='ecg_signal')
    # B.1 Conv
    dilation = [4, 4, 8, 16, 32, 64]

    X = Conv1D(64, 15, dilation_rate=dilation[0], strides=1, name='conv1
    ', kernel_initializer=glorot_uniform(seed=0))(ecg_input)
    X = BatchNormalization(name='bn_conv1')(X)
    X = Activation('relu')(X)
    X = MaxPooling1D(2, strides=2)(X)

    X = identity_block(X, 7, [64, 64], dilation=dilation[1], stage=2,
    block='a')
    X = identity_block(X, 7, [64, 64], dilation=dilation[2], stage=2,
    block='b')
    #(X, kernel, filters, dilation, stage, block, s=2)
    X = convolutional_block(X, 7, filters=[128, 128], stage=3, block='a'
    , s=2)
    X = identity_block(X, 7, [128, 128], dilation=dilation[3], stage=3, block='b')

    X = convolutional_block(X, 7, filters=[256, 256], stage=4, block='a', s=2)
    X = identity_block(X, 7, [256, 256], dilation=dilation[4], stage=4, block='b')


    X = convolutional_block(X, 7, filters=[512, 512], stage=5, block='a', s=2)
    X = identity_block(X, 7, [512, 512], dilation=dilation[5], stage=5, block='b')

    ecg_branch = tf.keras.layers.GlobalMaxPool1D()(X)
    # Concatenate
    shared_path = concatenate([ecg_branch, wide_branch], name='concat_layer')
    # HEAD Classifier
    shared_path = Dense(classes, activation='sigmoid', name='sigmoid_classifier')(shared_path)
    # Finally the model is composed by connecting inputs to outputs:
    model = Model(inputs=[ecg_input,  wide_input],outputs=shared_path)

    return model
```

Threshold Optimization

The threshold optimization algorithm performs an optimization of the thresholds by considering the Challenge Metric score obtained over the validation set. A direct comparison between the score obtained by fixing the thresholds at 0.5 and after the optimization is also performed by printing the results.

Listing A.5: Threshold Optimization algorithm

```
def threshold_optimization(name_valid_list, features_dict, deep_model, wide_model, redeep_model, selected_leads):
    '''
        Inputs:
          name_valid_list: list of subjects connsidered as validation set
          features_dict: dictionary of extracted wide features
          deep_model: trained deep model
          wide_model:trained wide+deep model
```

```python
          redeep_model: trained deep+wide+deep model
          selected_leads: leads subset considered
        Returns:
          deep_best_threshold: optimized threshold for the deep model
          deep_A: confusion matrix over the evaluation set of deep model
          wide_best_threshold: optimized threshold for the wide+deep model
          wide_A: confusion matrix over the evaluation set of wide+deep model
          redeep_best_threshold: optimized threshold for the deep+wide+deep model
          redeep_A: confusion matrix over the evaluation set of deep+wide+deep model

    '''
    print('Optimizing threshold...')
    ###STEP1###

    print('Extracting classes...')
    weights_file = './challengePackage/weights_new.csv'
    classes, weights = ec.load_weights(weights_file)
    sinus_rhythm = set(['426783006'])

    print("printing type of classes for debug...")
    #print(f'Checking that sinus rithm class is correct: {sinus_rhythm} is equal to 426783006?')

    # Get the lists for header and recording files
    header_files, recording_files = add_extension(name_valid_list)
    print('Extension added!')

    # load the true labels of the validation set
    labels = ec.load_labels(header_files, classes)
    print('True labels loaded!')
    #print(f"True labels: {(len(labels))}")
    ###STEP 2###
    # We use the for cycles from the first part of run_model to obtain the predictions and save them in a matrix

    sampling_freq = 500
    W = 5000
    # O = 256
    num_recordings = len(recording_files)

    # Initialize the matrix probability as an empty matrix

    deep_prob_saved = []
    wide_prob_saved = []
    redeep_prob_saved = []
    # Save the probabilities for each recording
    for i in tqdm(range(num_recordings)):
        # Load header and recording.
        header = load_header(header_files[i])
        current_signal = load_recording(recording_files[i])
        leads = get_leads(header)
        CH = len(selected_leads)
        # start_window = np.zeros((CH, W))
        # current_windows = []
        current_feature_dict = features_dict[name_valid_list[i]]
        current_features =  list(current_feature_dict.values())
        # Same preprocessing is applied to the test set
        # get_frequency is defined inside helper_code.py and provided by the challenge
        current_fs = get_frequency(header)
        current_sig_len = current_signal.shape[1]
        current_time_sec = current_sig_len / current_fs
        num_samples_new = int(current_time_sec * sampling_freq)
        current_signal = signal.resample(current_signal, num_samples_new, t=None, axis=1)

        #### STEP 2 #### Filtering and normalization
        filtered_ecg = pc.filtering_deep(current_signal, current_fs)
        rec_mean = np.nanmean(filtered_ecg, axis=1)
        rec_std = np.nanstd(filtered_ecg, axis=1)
        current_signal = pc.normalization_deep(filtered_ecg, rec_mean, rec_std)

        current_signal = mtc.subset_ch_recordings(leads, current_signal, selected_leads)
        reshaped_window = np.array(pc.extract_windows(current_signal, current_fs, W, CH, mode_use = 'train'))
        current_features = np.array(current_features)
        # predictions are performed on all the windows of the signal.
        deep_preds = deep_model.predict(reshaped_window)
        wide_preds = wide_model.predict([reshaped_window, np.array([current_features])])
        redeep_preds = redeep_model.predict([reshaped_window, np.array([current_features])])
        # the average is computed
        deep_probabilities = average_prediction(deep_preds)
        wide_probabilities = average_prediction(wide_preds)
        redeep_probabilities = average_prediction(redeep_preds)
        deep_prob_saved.append(deep_probabilities)
        wide_prob_saved.append(wide_probabilities)
        redeep_prob_saved.append(redeep_probabilities)


    #### STEP 3 ####
    print('CM of deep model with thr at 0.5...')
    ths_half = np.full(26, 0.5)
    deep_metric_no_opt, cnf_no_opt = ec.compute_challenge_metric(weights, labels, pred_to_binary(deep_prob_saved, ths_half), classes, sinus_rhythm)
    print(f'{deep_metric_no_opt}')

    print('CM of wide model with thr at 0.5...')
    ths_half = np.full(26, 0.5)
    wide_metric_no_opt, cnf_no_opt = ec.compute_challenge_metric(weights, labels, pred_to_binary(wide_prob_saved, ths_half), classes, sinus_rhythm)
    print(f'{wide_metric_no_opt}')

    print('CM of wide retrain deep model with thr at 0.5...')
    ths_half = np.full(26, 0.5)
    redeep_metric_no_opt, cnf_no_opt = ec.compute_challenge_metric(weights, labels, pred_to_binary(redeep_prob_saved, ths_half), classes, sinus_rhythm)
    print(f'{redeep_metric_no_opt}')

    # all attempts in the grid search
    possible_thrs = np.arange(0, 0.4, 0.01)
    # initialized vector of thresholds
    ths_dummy = np.full(26, 0.01)
    # Calculating metric for each threshold then selecting the threshold corresponding to the max value of the metric
    # Initialized list with selected threshold (26,)

    w, h = len(classes), len(possible_thrs)

    print('Deep optimization...')
    current_metric = np.zeros([w, h])
    # all_labels = np.array(all_labels)
    #print(possible_thrs)
    for thrs in tqdm(range(len(classes))):
        for t in range(len(possible_thrs)):
            ths_dummy[thrs] = possible_thrs[t]
            current_metric[thrs, t], A = ec.compute_challenge_metric(weights, labels, pred_to_binary(deep_prob_saved, ths_dummy), classes, sinus_rhythm)

    initialized_thr = np.zeros(len(classes))

    for i in tqdm(range(len(classes))):
        ix = np.argmax(current_metric[i, :])
        initialized_thr[i] = possible_thrs[ix]

    print(initialized_thr)

    deep_best_threshold = fmin(thr_to_challenge_metric, args=(weights, classes, sinus_rhythm, labels, deep_prob_saved), x0=initialized_thr)
    #best_threshold = minimize(thr_to_challenge_metric, x0 = initialized_thr, args=(weights, classes, sinus_rhythm, labels, prob_saved),
    #  method='L-BFGS-B', options={'disp': None, 'maxcor': 10, 'ftol': 2.220446049250313e-09, 'gtol': 1e-05, 'eps': 1e-08,
    #  'maxfun': 15000, 'maxiter': 15000, 'iprint': 0, 'maxls': 100, 'finite_diff_rel_step': None})
    print('The deep model threshold vector is:')
    deep_current_metric, deep_A = ec.compute_challenge_metric(weights, labels, pred_to_binary(deep_prob_saved, deep_best_threshold), classes, sinus_rhythm)
    print(deep_best_threshold)
    print(f'the deep model current metric is: {deep_current_metric}')

    print('Wide optimization...')
    current_metric = np.zeros([w, h])
    for thrs in tqdm(range(len(classes))):
        for t in range(len(possible_thrs)):
            ths_dummy[thrs] = possible_thrs[t]
            current_metric[thrs, t], A = ec.compute_challenge_metric(weights, labels, pred_to_binary(wide_prob_saved, ths_dummy), classes, sinus_rhythm)

    initialized_thr = np.zeros(len(classes))

    for i in tqdm(range(len(classes))):
```

```python
            ix = np.argmax(current_metric[i, :])
            initialized_thr[i] = possible_thrs[ix]

    print(initialized_thr)

    wide_best_threshold = fmin(thr_to_challenge_metric, args=(weights, classes, sinus_rhythm, labels, wide_prob_saved), x0=initialized_thr)
    #best_threshold = minimize(thr_to_challenge_metric, x0 = initialized_thr, args=(weights, classes, sinus_rhythm, labels, prob_saved),
    #  method='L-BFGS-B', options={'disp': None, 'maxcor': 10, 'ftol': 2.220446049250313e-09, 'gtol': 1e-05, 'eps': 1e-08,
    #  'maxfun': 15000, 'maxiter': 15000, 'iprint': 0, 'maxls': 100, 'finite_diff_rel_step': None})
    print('The wide threshold vector is:')
    wide_current_metric, wide_A = ec.compute_challenge_metric(weights, labels, pred_to_binary(wide_prob_saved, wide_best_threshold), classes, sinus_rhythm)
    print(wide_best_threshold)
    print(f'the wide current metric is: {wide_current_metric}')

    print('Wide with retrain deep optimization...')
    current_metric = np.zeros([w, h])
    # all_labels = np.array(all_labels)
    #print(possible_thrs)
    for thrs in tqdm(range(len(classes))):
        for t in range(len(possible_thrs)):
            ths_dummy[thrs] = possible_thrs[t]
            current_metric[thrs, t], A = ec.compute_challenge_metric(weights, labels, pred_to_binary(redeep_prob_saved, ths_dummy), classes, sinus_rhythm)

    initialized_thr = np.zeros(len(classes))

    for i in tqdm(range(len(classes))):
        ix = np.argmax(current_metric[i, :])
        initialized_thr[i] = possible_thrs[ix]

    print(initialized_thr)

    redeep_best_threshold = fmin(thr_to_challenge_metric, args=(weights, classes, sinus_rhythm, labels, redeep_prob_saved), x0=initialized_thr)
    #best_threshold = minimize(thr_to_challenge_metric, x0 = initialized_thr, args=(weights, classes, sinus_rhythm, labels, prob_saved),
    #  method='L-BFGS-B', options={'disp': None, 'maxcor': 10, 'ftol': 2.220446049250313e-09, 'gtol': 1e-05, 'eps': 1e-08,
    #  'maxfun': 15000, 'maxiter': 15000, 'iprint': 0, 'maxls': 100, 'finite_diff_rel_step': None})
    print('The wide with retrain deep model threshold vector is:')
    redeep_current_metric, redeep_A = ec.compute_challenge_metric(weights, labels, pred_to_binary(redeep_prob_saved, redeep_best_threshold), classes, sinus_rhythm)
    print(redeep_best_threshold)
    print(f'the deep model current metric is: {redeep_current_metric}')

    return deep_best_threshold, deep_A, wide_best_threshold, wide_A, redeep_best_threshold, redeep_A
```

Ensemble Evaluation

The algorithm for the ensemble evaluation performs a processing of test data which is the same one used in training, then loads all the models and makes predictions with majority voting.

Listing A.6: Ensemble Evaluation algorithm

```
def ensemble_evaluation(all_models, all_json, header, recording, thr_opt_models, json_directory):
    '''
        This function extracts and performs the mean value of the probability obtained in the test from all the models
        Should be iteratively applied on a single header-recording couple each time. Used in run_model()

        Inputs:
            all_models: list() containing all LOADED models
            all_classes: list() containing all LOADED json files corresponding to the respective model
            optimal_thr_models: np.array containing the extracted mean value of threshold of models
            header: of the signal
            recording: of the signal
        Returns:
            signal_probs: np.array with the MEAN of all the probabilities estimated from the models taken into account
            classes: 26 classes taken from json dictionary
            labels: predicted labels of the signal
    '''
    W = 5000
    resampling_freq = 500
    pickle_filename = os.path.join(json_directory, 'minmax_scaler.pkl')
    # with open(pickle_filename, 'rb') as file:
    #     fitted_scaler = pickle.load(open(pickle_filename))
    fitted_scaler = pickle.load(open(pickle_filename, 'rb'))

    labels = np.zeros(shape=(len(all_models), 26))

    leads = get_leads(header) #seguente parte messa fuori cos  la esegue una sola volta per segnale e non ad ogni iterazione del for per ogni modello
    CH = len(leads)
    current_fs = get_frequency(header)
    current_sig_len = get_num_samples(header)
    current_time_sec = current_sig_len / current_fs
    num_samples_new = int(current_time_sec * resampling_freq)
    current_signal = signal.resample(recording, num_samples_new, t=None, axis=1)
    ## STEP 0 - Signal filtering and normalization deep ##
    filtered_ecg = pc.filtering_deep(current_signal, current_fs)
    rec_mean = np.nanmean(filtered_ecg, axis=1)
    rec_std = np.nanstd(filtered_ecg, axis=1)
    current_signal = pc.normalization_deep(filtered_ecg, rec_mean, rec_std)
    current_signal = mtc.subset_ch_recordings(leads, current_signal, leads)
    reshaped_window = np.array(pc.extract_windows(current_signal, current_fs, W, CH, mode_use = 'test'))


    #### STEP 1 - WIDE FEATS ####
    # Extract windows from recording to be predicted
    my_leads = ('II')
    # STEP. Preprocessing of the signal before filtering.
    # print("the leads obtained from header are:")
    # print(leads)
    ecg_ord = mtc.subset_ch_recordings(leads, recording, my_leads)
    #print(len(ecg_ord))
    # STEP. Filtering and normalization
    try:
        signal_properties = biosppy.signals.ecg.ecg(ecg_ord[0], sampling_rate=current_fs, show=False)
        rpeaks = signal_properties['rpeaks']
        nnintervals = tools.nn_intervals(rpeaks)
        # STEP. Feature Extraction
        current_feature = pd.DataFrame()

        if len(nnintervals) > 1:
            # If R peaks are at least 2, we extract features
            time_features, num_keys = pc.time_domain_features(nnintervals, 'prediction_code')
            # STEP 10. Poincar  plot features
            pc_features, numerical_columns = pc.new_features(nnintervals, rpeaks, 'prediction_code')
            numerical_columns = numerical_columns[1:]
            num_keys = num_keys + numerical_columns
            feat_dict = {**time_features, **pc_features}
            # STEP 11. Nonlinear features - INFINITE VALUES here too
            del feat_dict['poincare_plot']
            # STEP 8. Append to Dataframe in every case
            current_feature = current_feature.append(feat_dict, ignore_index=True)
        else:
            print(f'Only 1 R peak detected... NaNs returned')
            feat_dict = {
                'mean_nni': np.NaN,
                'sdnn': np.NaN,
                'sdsd': np.NaN,
                'nni_50': np.NaN,
                'pnni_50': np.NaN,
                'nni_20': np.NaN,
                'pnni_20': np.NaN,
                'rmssd': np.NaN,
                'median_nni': np.NaN,
                'range_nni': np.NaN,
                'cvsd': np.NaN,
                'cvnni': np.NaN,
                'mean_hr': np.NaN,
                "max_hr": np.NaN,
                "min_hr": np.NaN,
                "std_hr": np.NaN,
                'name' : 'prediction_code',
                'sd1' : np.nan,
                'sd2' : np.nan,
                'sd_ratio' : np.NaN,
                'ellipse_area' : np.NaN
            }
    except:
        current_feature = pd.DataFrame()
        print(f'No R peaks detected... NaNs returned')
        feat_dict = {
                'name' : 'prediction_code',
                'mean_nni': np.NaN,
                'sdnn': np.NaN,
                'sdsd': np.NaN,
                'nni_50': np.NaN,
                'pnni_50': np.NaN,
                'nni_20': np.NaN,
                'pnni_20': np.NaN,
                'rmssd': np.NaN,
                'median_nni': np.NaN,
                'range_nni': np.NaN,
                'cvsd': np.NaN,
                'cvnni': np.NaN,
                'mean_hr': np.NaN,
                "max_hr": np.NaN,
                "min_hr": np.NaN,
                "std_hr": np.NaN,
                'sd1' : np.nan,
                'sd2' : np.nan,
                'sd_ratio' : np.NaN,
                'ellipse_area' : np.NaN
            }
        current_feature = current_feature.append(feat_dict, ignore_index=True)
        num_keys = list(feat_dict.keys())[1:]

    current_feature =  current_feature.set_index('name')
    current_feature.fillna(-1, inplace=True)

    #mins = [np.nanmin(current_feature.values[:, i][current_feature.values[:, i] != -np.inf]) for i in range(current_feature.shape[1])]
    #maxs = [np.nanmax(current_feature.values[:, i][current_feature.values[:, i] != np.inf]) for i in range(current_feature.shape[1])]

    # go through matrix one column at a time and replace  + and -infinity
    # with the max or min for that column
```

```
    for i in range(current_feature.shape[1]):
        current_feature.values[:, i][current_feature.values[:, i] == -np.inf] = 0
        current_feature.values[:, i][current_feature.values[:, i] == np.inf] = 99999

        # print(current_feature[:].values)

    current_feature[:] = fitted_scaler.transform(current_feature[num_keys])
    feature_dict = current_feature.to_dict('index')
    feats = list(feature_dict['prediction_code'].values())
    # print(f"the leads I want to select are: {dictionary_model['leads']}")

    pred_feats = []
    for i in range(len(reshaped_window)):
        pred_feats.append(np.array(feats))

    for model in range(len(all_models)):
        #print(all_json)
        # print(all_json[model])
        # Load info of json file of current model
        dictionary_model = all_json[model]
        # need to define varying number of leads depending on model.
        #CH = len(dictionary_model['leads'])

        selected_leads = dictionary_model['leads']
        # print("printing dictionary model")
        # print(dictionary_model['leads'])
        classes = dictionary_model['classes']


        #### STEP 2 ####
        # predictions are performed on all the windows of the signal.
        #preds = all_models[model].predict(np.array(reshaped_window))
        preds = all_models[model].predict([np.array(reshaped_window), np.array(pred_feats)])
        #### STEP 3 ####
        probabilities = average_prediction(preds)
        # print(f'The model {model + 1} performs the following probabilities on the signal: {probabilities}')
        # We take the mean probability of the prediction of all models
        #signal_probs[model, :] = probabilities
        labels[model, :] = pred_to_binary(probabilities, thr_opt_models[model, :])

    probabilities = np.sum(labels , axis=0)/3 #(labels[0] + labels[1] + labels[2])/3 #c'   probabilmente modo migliore per selezionare le prob che 0,1,2.. d   prob secondo ensemble quindi 0,0.3,0.6 e 1
    labels = (probabilities>0.5)*1    #per avere poi label binarizzate. se 2 su 3 danno 1 prob = 0.6

    #signal_probs = np.mean(signal_probs, axis=0)
    ## print(f'After the average the probability results as: {signal_probs}')
    #labels = pred_to_binary(signal_probs, thr_opt_models)

    return classes, labels, probabilities
```

List of Figures

1.1 Perceptron Unit . . . 5
1.2 Binary Activation Functions . . . 6
1.3 Continuous nonlinear Activation Functions . . . 6
1.4 Unbounded Activation Functions . . . 7
1.5 Multi-Layer Perceptron . . . 7
1.6 Feed-forward Neural Network . . . 8
1.7 Feedback Neural Network . . . 9
1.8 Recurrent Neural Network . . . 10
1.9 CNN - Information Abstraction and Compression . . . 11
1.10 CNN - Feature Maps . . . 12
1.11 CNN - Convolution . . . 13
1.12 CAM in CNN . . . 14
1.13 Supervised and Unsupervised learning . . . 16
1.14 Error tradeoff . . . 18
1.15 Data processing . . . 19
1.16 Model complexity . . . 19
1.17 Ensemble of models . . . 21

2.1 ECG template . . . 25
2.2 ECG acquisition scheme . . . 26
2.3 Bidirectional data processing . . . 33
2.4 Network architecture . . . 36
2.5 Residual blocks architecture . . . 37
2.6 FCNN bottleneck . . . 38
2.7 Original label distribution . . . 41
2.8 Label distribution for ensemble . . . 42
2.9 Ensemble classifier . . . 43

3.1 CM weights . . . 45
3.2 12-leads deep model Confusion Matrix . . . 52

3.3 2-leads deep model Confusion Matrix 53
3.4 12-leads wide model Confusion Matrix 54
3.5 2-leads wide model Confusion Matrix 55
3.6 12-leads finetuning model Confusion Matrix 56
3.7 2-leads finetuning model Confusion Matrix 57

List of Tables

2.1 Label distribution per database 29
2.2 Label nature classification 32
2.3 List of temporal parameters 35

3.1 Unofficial Phase results 47
3.2 Official Phase results 48
3.3 Focus Issue results 49
3.4 Local test set results 50
3.5 Positive Predictive Values 51