CARDIAC ARRHYTHMIA CLASSIFICATION USING STACKED ENSEMBLE TECHNIQUES.

Jordan Morris
Student ID: 10233834
The University of Manchester
Department of Physics and Astronomy
(This experiment was performed under the Supervision of Henggui Zhang and in collaboration with Jonathan Ormrod.)

(Dated: May 15, 2023)

A cardiac arrhythmia is caused due to a disruption in one of the seven cardiac action potentials present in the heart [1]. They are visualised using electrocardiograms (time, voltage graphs) whose signals are extraordinarily complex. It is this complexity that makes them difficult to classify for physicians and is the primary motivation for the use of machine learning. The CPSC-2018 dataset provides access to open-source 12-lead ECG data consisting of 6,877 patients [2]. This data was processed using a bandpass filter, the Pan-Tompkins algorithm, and continuous wavelet transforms (CWT). From the 2D scalograms produced by the CWTs, 12 independent convolutional neural networks were trained, thus constituting a level zero space of a stacked ensemble model. These each produced F1 scores in the range of $0.85-0.90$, with the exception of three leads (V2, V4, V6). These leads appear to not be well suited to CWTs and were therefore excluded from the ensemble model. In order to construct the level one space, a meta-learning model was used. Several different models were trailed, a 1D CNN meta-model performed best and achieved an F1 score of 0.9776. Overall this meta-model perfectly predicted three of the nine classes and achieved a score of $F_{1i} \geq 0.95$ on all other classes.

I. INTRODUCTION

A. Cardiac action potentials and arrhythmia

The heart is an electromechanical pump whose function depends on the action potentials generation and propagation [1]. The myocardial tissue is comprised of two tissue types: ventricular and pacemaker [3]. These each exhibit independent action potentials, which reflect the sequential activation and inactivation of various ion channels [1, 3]. There are seven action potentials in the heart consisting of five ventricular and two pacemaker potentials [1].

The ventricular action potential is a 5-phase sequential process that corresponds to the QT interval seen on Figure 1 [1, 3, 4]. It depends on the Na^+, Ca^{2+}, and K^+ ion channels [3]. The Na^+ channel initiates ventricular depolarisation until the channel closes, at which point the Ca^{2+} channel produces further depolarisation [3]. This is then balanced by the repolarising K^+ channel and for a time the action potential is in equilibrium at $30mV$ [3]. As the Ca^{2+} channel closes the K^+ channel begins to dominate resulting in a rapid repolarisation toward the resting potential, $-80mV$ [3].

The pacemaker nodes exhibit automaticity, therefore, they can beat spontaneously without stimulus [5]. This can be attributed to pacemakers not having a true resting potential. At the start of a cycle the membrane potential increases towards positive values due to Na^+ and K^+ currents [5]. This is followed by Ca^{2+} dominated depolarisation until K^+ dominates and repolarises the membrane [3, 5]. Therefore, the pacemaker action potential is a 4-phase process, it acquires its namesake, 'pacemaker', through the faster beat rate of the nodal cells (this sets the pace of the heart) [3, 5].

A cardiac arrhythmia manifests as a disruption of one or more action potentials. The many different action potentials and the complex systems governing them indicate why there are so many different forms of arrhythmia [6]. This study shall consider sinus rhythm (SR) and eight types of arrhythmia: atrial fibrillation (AF), first-degree atrial ventricular block (I-AVB), left bundle branch block (LBBB), right bundle branch block (RBBB), premature atrial contractions (PAC), premature ventricular contractions (PVC), ST-depression (STD), and ST-elevation (STE). Note, what these arrhythmia actually are is not pertinent, what is however is that each will manifest in a unique form on an electrocardiogram.

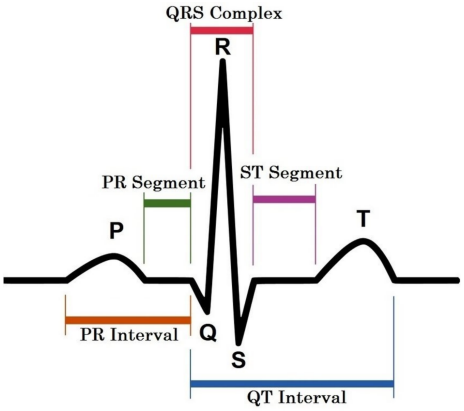

FIG. 1. A heartbeats waveform as presented on an ECG [7].

B. Electrocardiogram structure

In order to test for cardiac arrhythmia, electrocardiograms (ECG) have been used for over a century [4, 6]. ECGs are quick, non-invasive, and can be used in various aspects of biomedical science [8]. Through the use of 12 leads, placed strategically on the body, an ECG machine is able to record the electrical activity of the heart and produce a time, voltage graph. These graphs can be broken down into individual waveforms, which in turn can be further deconstructed into the segments and points seen in Figure 1. When an arrhythmia is present, it will change the inherent structure of one or more sections of an ECG. An example of how an arrhythmia can affect an ECG can be seen in Figure 2.

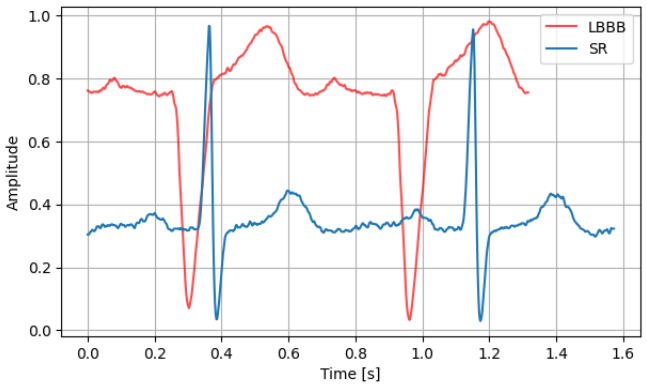

FIG. 2. Two patients (both lead V5) from the CPSC-2018 dataset were segmented to two beats. The patient in sinus rhythm is in blue, whilst the arrhythmic patient (left bundle branch block) is in red. These serve as an example of the difference an arrhythmia can make to an ECG.

C. Motivating the use of machine learning

In order to classify cardiac arrhythmia in a reliable manner, years of training and experience are prerequisite, this is due to the complex nature of ECG signals [9]. However, even after many years of training, they can still be misdiagnosed [9]. When considering that this could be the difference between life or death for a patient, more must be done to support cardiologists. In order to assist physicians, machine learning algorithms may be developed to provide quick, automatic, and accurate diagnoses [9].

There are often two approaches to this task: beat-based and end-to-end analysis [9]. The beat-based analysis is where a signal is divided into single or multi-beat segments [9]. The advantage of this approach is it requires very few patient recordings, however, one must take care to avoid temporal leakage or they may accidentally produce false results [9]. Such was the case in Qin et al. [9, 10]. The end-to-end form of analysis requires a large dataset and an appropriate deep learning algorithm to take advantage of it [9]. This study shall look into taking advantage of both techniques.

II. CHINESE PHYSIOLOGICAL COMPETITION DATASET

The dataset used for this study was the Chinese physiological signal challenge (CPSC) from the 2018 competition. This was chosen as it is open source and has many competition entries available for direct result comparisons [2]. The database provides a training set of 6,877 entries and a validation set of 300 entries, however, they do withhold their testing set [2]. To counter this, the validation set provided by CPSC was used as a pseudo-testing set. In doing this it can be guaranteed that there is no leakage between training and testing. Both of these datasets were provided with an associated '.csv' files which contained patient names and labels. Each patient can have up to three labels, therefore constituting a multi-label problem[1].

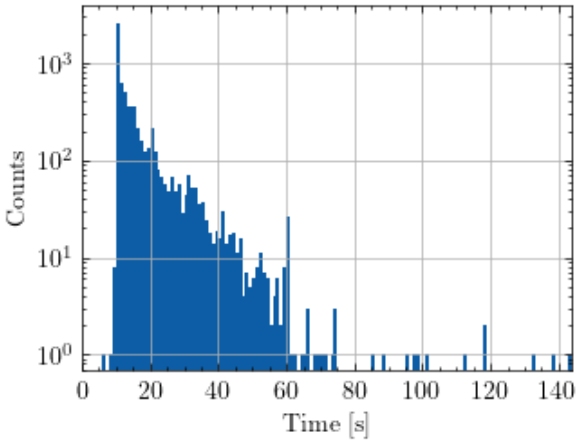

FIG. 3. A histogram of the time distribution throughout the CPSC-2018 database. Note most data entries exist at $10s$ and that there are entries of length $> 60s$.

Each data entry possesses all 12 leads of ECG stored in a '.mat' file. These entries are of variable length with a minimum of $6s$ and a maximum of $60s$, as per the competition's website [2]. These statements were double-checked and thus produced Figure 3. The histogram displays the minimum time length to be at $6s$, however, the maximum lies well above $60s$. This is in contradiction to the competition website as it explicitly states that the

[1] Note there is a significant difference between multi-class and multi-label problems. A multi-class problem is where each entry may take on one class value, y_i, out of a set of N classes, $y_i \in [y_1, y_N]$. A multi-label problem is where each entry may take on several class values, $\boldsymbol{y} = (y_i, ..., y_j)$, where $y_i, ...y_j \in [y_1, y_N]$.

maximum recording length is 60s. Therefore, any recordings found to be greater than 60s in length were excluded from this study.

Further patients were also removed from this study if any of the leads had a mean reading of zero as per last semester [4].

Asserting both of these requirements onto the dataset resulted in the 6,877 patients being reduced to 5,738 patients, a more detailed breakdown of what arrhythmia classes lost the most data can be seen in Table I. Expressed as a percentage, the total loss of training data was 16.56%. This loss, whilst significant, will easily be countered by the artificial boosting due to the sliding window routine - see Section III B.

	Before Checking		Post Checking	
Arrhythmia	**Primary**	**Multi-Label**	**Primary**	**Multi-Label**
SR	907 (45)	0 (0)	772 (44)	0 (0)
AF	1081 (43)	121 (4)	911 (43)	94 (4)
I-AVB	695 (25)	17 (1)	592 (25)	13 (1)
LBBB	202 (11)	27 (1)	167 (11)	23 (1)
RBBB	1677 (71)	159 (3)	1422 (71)	134 (3)
PAC	566 (26)	41 (2)	474 (26)	35 (2)
PVC	643 (30)	47 (1)	542 (30)	35 (1)
STD	814 (33)	41 (3)	677 (33)	33 (3)
STE	200 (16)	15 (0)	181 (16)	12 (0)
Total Patients	6877 (300)		5738 (299)	

TABLE I. The number of data entries for each class in the CPSC-2018 dataset as well as the number of them that are multi-labelled. The initial distribution is before running the 'valid data' checks described previously and the final one is after running these checks. Note the values in brackets are for the test dataset.

III. DATA PRE-PROCESSING METHODS

A. Bandpass filtering complex signals

ECGs are complex signals containing many different forms of low ($\nu < 1$Hz) and high-frequency ($\nu > 1$Hz) noise [11, 12]. The list of potential sources is extensive, some examples are motion artefacts, electrical interference, and respiration [11, 12]. To counteract these noise components a bandpass filter was employed.

A bandpass filter is characterised by its transfer function $H(\nu)$, where this can be dependent on different types of filters [13]. This study makes use of two Butterworth filters, high-pass (H) and low-pass (L). Butterworth filters are designed to be maximally flat in the passband region, for example at extremely high orders the bandpass function seen in Figure 4 will tend towards a step function [11]. The Butterworth high-pass filter obeys

$$H_H(\nu) = \frac{1}{\sqrt{1 + (\frac{\nu_c}{\nu})^{2n}}}, \tag{1}$$

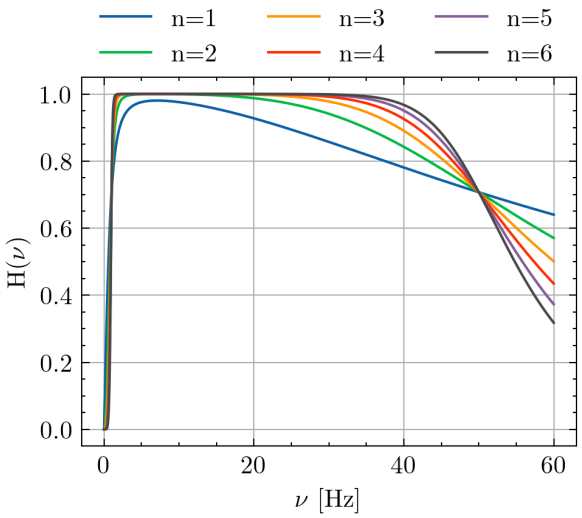

FIG. 4. A plot of the a bandpass filter with low-cut, 1Hz and high-cut, 50Hz. The filter order was allowed to vary up to six. Notice that the curves become flatter for longer within the bandpass range and more high-frequency components are cut off at higher orders.

where ν is the frequency, ν_c is the specified low-cut frequency, and n is the filter order [14]. Similarly, the low-pass Butterworth filter can be written as

$$H_L(\nu) = \frac{1}{\sqrt{1 + (\frac{\nu}{\nu_c'})^{2n}}}, \tag{2}$$

where ν_c' is the high-cut frequency [14]. The bandpass function is the product of these two functions and is therefore described via

$$H(\nu) = H_H(\nu)H_L(\nu). \tag{3}$$

An example of $H(\nu)$ can be seen in Figure 5, here the passband exists on the interval $[1, 50]$Hz. These values were chosen as this is approximately the useful region of data [11]. Explicitly the 'useful' region is $0.25 \leq \nu \leq 50$ Hz due to the data being sampled at 500Hz [11]. The filter order, n, was decided through experimentation.

Seen in Figure 4, is the bandpass function where the filter order is allowed to vary (the high and low-cut parameters are as above). As the filter order increases, the low-cut of 50Hz slowly becomes more aggressive, therefore, more data is allowed to pass through in the passband. This can be seen with the bandpass peak becoming flatter with increasing n. Some literature suggests an ideal set of parameters is a passband of $8 - 20$Hz (high-cut, low-cut) with a filter order of 8 [12]. The passband parameters suggested cannot be converted as [12] does not provide the sampling frequency of their data. However, a filter order of 8 was trailed, which resulted in a complete loss of signal. The ECG recording thus became a single flat line. In order to preserve the signal a filter

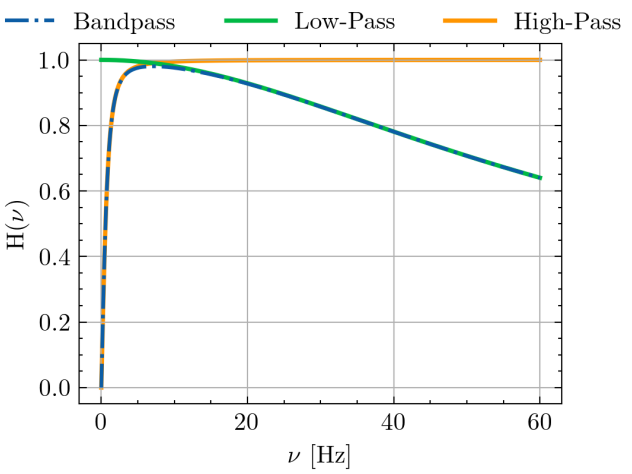

FIG. 5. A Butterworth bandpass filter and its two components, the high and low-pass. In order to form the bandpass these two are multiplied. This is an example of the bandpass used in this study of high-cut 1Hz, low-cut 50Hz, and filter order one.

order of $n = 1$ was chosen, this can be seen by comparing Figure 4 and 5. Filter orders two and three were also considered, however, these appeared to alter the signal such that extra components were generated. Using a filter order of two appeared to generate an extra P-wave in most, if not all samples.

Once processed all signals were normalised such that the amplitude existed on the interval $[0, 1]$.

B. The Pan-Tompkins and sliding window algorithms

In the previous semester, the study concluded that one method hindering the accuracy of the results was the segmentation routine [4]. Therefore, this was redeveloped from the ground up.

When developing the sliding window, the primary aim was for each segment to contain the same number of heartbeats. Therefore, a sliding window based on the time series data alone was too simplistic. To combat this the Pan-Tompkins algorithm was used for R peak detection. The Pan-Tompkins algorithm is a two-stage process - pre-processing and peak detection [15].

The pre-processing stage consists of four steps. Initially, the data is passed through a bandpass function of passband $0.01 - 15$Hz and filter order 1, the product of which can be seen in Figure 6.1. These parameters were chosen as a passband in the 5-35Hz maximises QRS energy allowing for superior overall identification [16]. The next step in this process is differentiation [16]. Using differentiation allows for the identification of the large gradients that are commonly associated with QRS complexes [16]. The process of differentiating also suppresses the low-frequency components such as the P and T waves, further exaggerating the QRS complex [16]. This produces Figure 6.2. The result of this is then squared, further enhancing the difference between the small structures and the QRS complex - see Figure 6.3 [16]. Finally, the squared waveform is passed through an integration window. The integrated signal, $y(nT)$, is calculated as

$$y(nT) = \frac{1}{N}\sum_{i=1}^{N}[x(nT - (N - i)T)], \tag{4}$$

where N is the number of samples contained within the window width and $x(nT - (N - i)T)$ is the input ECG signal as a function of the sample period T and n, $n \in \mathbb{Z}$ [15, 17]. The integration window width is set to $150ms$ as per [15]. The results of this can be seen in Figure 6.4.

Using this data allows for the second stage to begin, peak detection. The purpose of this is to classify the integration window results and determine whether it is or is not a QRS complex [15]. For this study, the baseline limit was set to one-half of the mean values in the data produced by the integration window. An example of this peak detection can be seen in Figure 6.4, where the detected peaks are marked with red asterisks overlaying the integrated signal. These peak positions may then be passed to the sliding window algorithm.

The sliding window is able to capture a segment of n peaks, n may be varied but for this study, it was fixed as $n = 3$. Each segment has an overlap of $n - 1$ peaks, the implementation here results in a two-beat overlap. Therefore, segment $N + 1$ will contain the last two peaks of segment N, thereby temporally linking the segments. An example of a single segment is indicated in Figure 6.5, where the red vertical lines represent the start and end of the second segment for that signal. Using this method allowed for an artificial boosting in the dataset taking the 5,738 valid patients, to 119,085 total segments.

Using the segments produced by the sliding window, it was found that the most abundant classes were being over-predicted by the neural network. This produced a vast amount of false positive predictions, significantly harming the final results. In order to combat this under-sampling was implemented.

C. Under-sampling and class weighting methods

Under-sampling is a technique whereby data points are randomly removed from a dataset to distribute the data more evenly [18]. This allows for techniques such as class weighting to be more effective. Seen in Figure 7 is an example of how the data distribution is affected by under-sampling. Any data exceeding a pre-determined value, $\mathcal{N}$, was reduced to approximately this value. This is only approximate due to the stochastic nature of the process used.

The selection process is dictated by a selection proba-

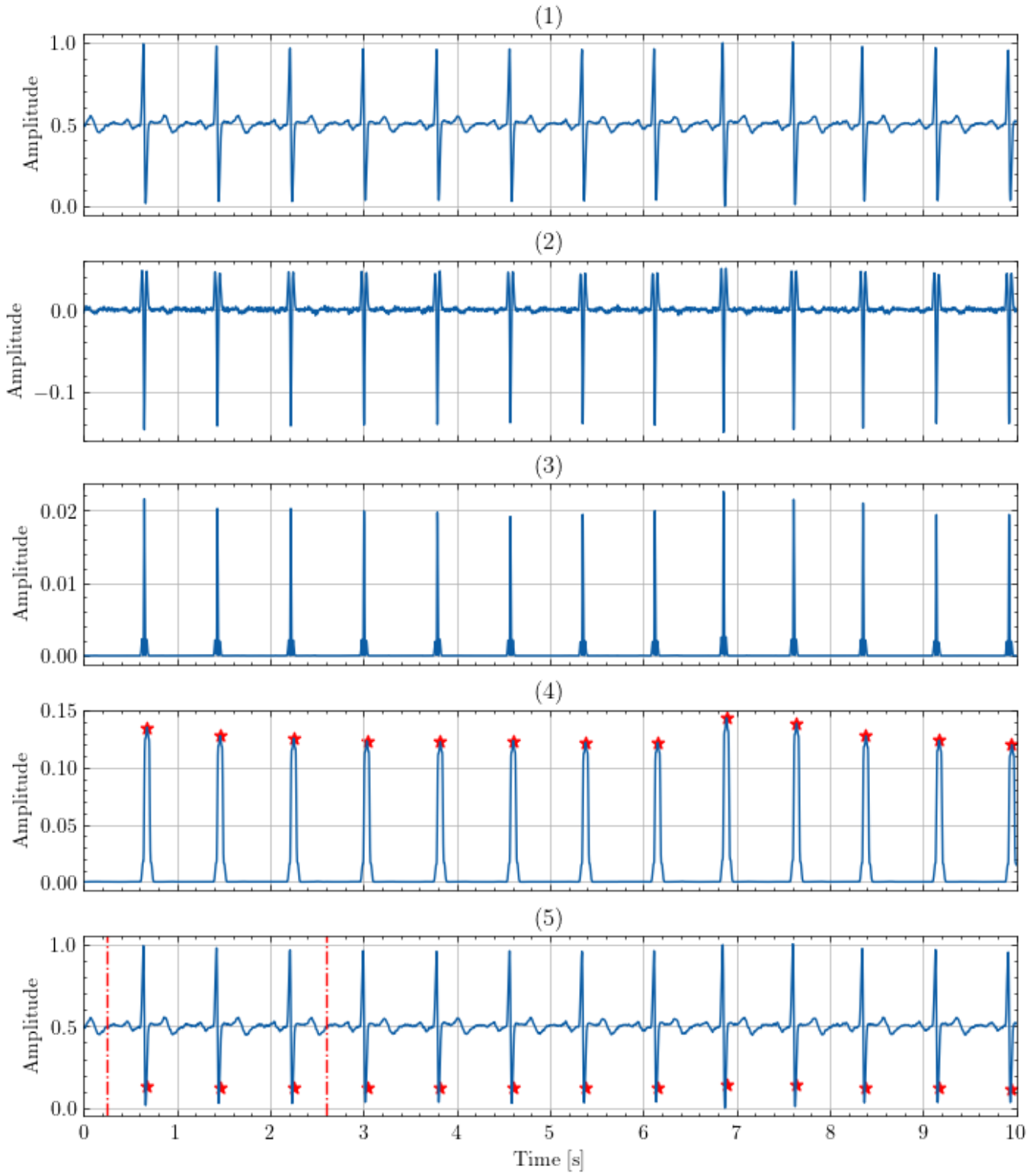

FIG. 6. An example of the Pan-Tompkins algorithm being applied to lead V5 for a patient in sinus rhythm. The steps are labelled in order of application: 1, bandpass filtering, 2, differentiation, 3, squaring, and 4, the integration window. Figure 6.4 shows the integrated signal along with the detected peaks (marked with red asterisks). Finally, Figure 6.5 displays the original signal with the peak detections still marked as well as an example segment that is detected by the sliding window routine.

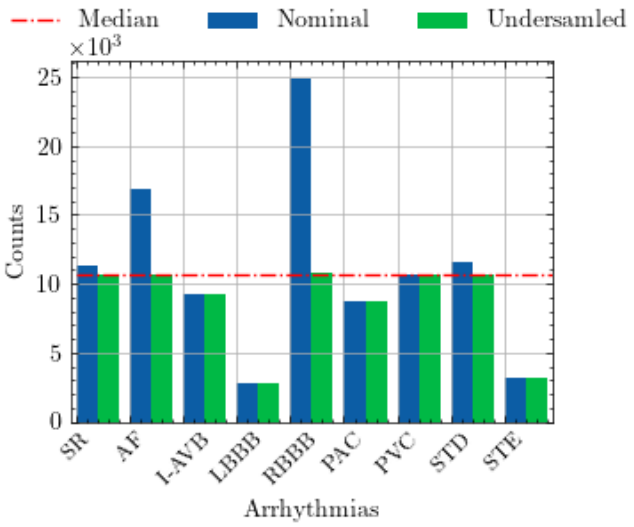

FIG. 7. The total number of segments produced for each class is seen in blue. An example limit is shown in red (the median). In green is the undersampled data distribution, each class exceeding the median is reduced to approximately that value.

bility, P_i, that is calculated via

$$P_i = \frac{\mathcal{N}}{N_i}, \tag{5}$$

N_i is the number of counts present in class i. For the purpose of this study $\mathcal{N} = 12000$, this decision was influenced by the size of the GPU used[2]. In the case where $P_i < 1$, the class will be reduced using a random uniform distribution. For every data point in this case a random variable, $\mathcal{R}$, which exists on the interval $[0, 1]$ is generated. If $P_i \geq \mathcal{R}$ then the data point is retained and is added to the under-sampled data. However, if the opposite is true and $P_i < \mathcal{R}$ then the data point is discarded completely. In the opposite case where $P_i \geq 1$, i.e. the total number of samples in class i is less than $\mathcal{N}$, every sample in that class is guaranteed to be retained. This can be seen clearly with the LBBB and STE classes in Figure 7. Using a stochastic method is important as for under-sampling to be truly effective the selection of samples must be random [18].

Finally, before this new under-sampled dataset $\mathcal{D}'$ is passed to the network, the class weightings are calculated. This method was retained from last semester and the class weightings are calculated via

$$\Omega_i = \frac{\sum_{j=1}^{N_c} \chi_j}{\chi_i}, \tag{6}$$

[2] Note that $\mathcal{N}$ was initially set to the median value of the dataset but the GPU crashed due to the RAM limitations. Therefore, $\mathcal{N}$ was reduced to 12000.

χ_i is the number of samples that belong to class i, where $i \in [1, N_c]$ [4].

D. Continuous wavelet transforms

In order for a neural network to classify ECGs a feature extraction method must first be decided on. This study makes use of continuous wavelet transform (CWT) to convert the signals to time-frequency space [19]. Given a signal $x(t)$, a CWT is defined as

$$C_a(b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi(\frac{t-b}{a})dt, \tag{7}$$

where a is a scale parameter, b, a translation parameter, and $\psi(t)$ is the mother wavelet function [19]. The mother wavelet chosen for this study was the Mexican Hat wavelet as it is well suited to ECG analysis [19]. The Mexican Hat wavelet is given as

$$\psi(t) = \frac{2}{\sqrt{3}\sqrt[4]{\pi}} \exp(-\frac{t^2}{2})(1-t^2), \tag{8}$$

where the preceding constant is a normalisation factor and t, time [4, 19, 20]. Once the ECG signal had been converted to time-frequency space, the data was resized to a 128x128 format. This was so a square scalogram was produced which is more suitable for a 2D CNN, the product of this can be seen in Figure 8.

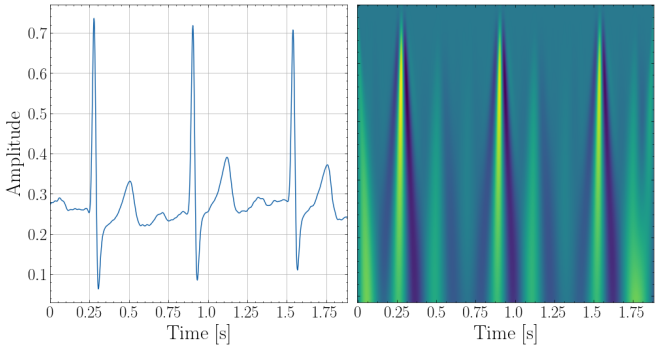

FIG. 8. An ECG signal extracted from the CPSC dataset which was sampled at 500Hz, this signal has been subject to the previously mentioned pre-processing techniques. Next to it is its CWT counterpart, this was calculated using the PyWavelet python package and then resized to a 128x128 image [20].

IV. MODEL COMPONENTS

A. An introduction to convolution neural networks

An artificial neural network (ANN) aims to be a computerised representation of a biological nervous system

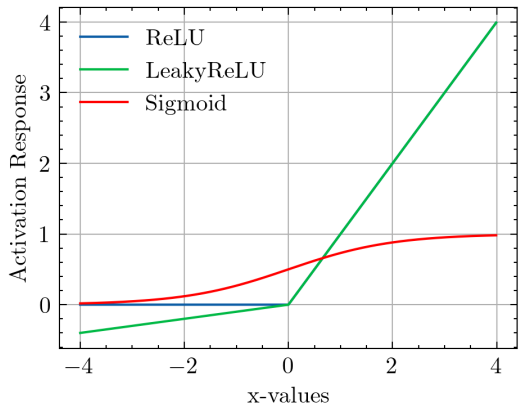

FIG. 9. An example of three activation functions: ReLU, LeakyReLU, and sigmoid. LeakyReLU is a modification on the ReLU function such that $\forall x \leq 0 : y = 0.1x$.

and is therefore often discussed in terms of neurons and neuron connections (synapses) [21]. Mathematically a neuron, y, may be represented via,

$$y = \theta(\sum_{i=1}^{n} w_i x_i + w_0), \tag{9}$$

CNNs are designed to handle tensor-like data and typically contain convolution, pooling, and dense (fully connected) layers [22, 23]. The convolution layer does exactly what its name states, it convolves the input with some kernel [22]. In doing this a CNN produces a feature map (F), which is then passed to a non-linear activation function. There are many types of activation functions, ReLU, LeakyReLu, Sigmoid, and tanh are just some of the more common ones [24]. These functions exist to represent the potentially non-linear and highly complex functions that may map the input to the output [24]. Once activated, the feature map is often passed to pooling layers. There are two primary forms: maximum (max) and average pooling. As per their namesake, max pooling returns the maximum value of some region in the feature map, whereas, average pooling returns the average. In doing this a pooling layer down-samples a feature map, producing F', which is approximately invariant to small translations of the input [22]. This is particularly useful when the motivation is to detect a feature, not necessarily where this feature is [22]. From there a CNN may flatten the feature maps and pass them to a fully connected layer. This final layer produces a probability distribution of the network's class prediction.

Described above is a rather simplistic CNN, in practice it is often more complicated. Deep neural networks (including CNNs) often suffer from over-fitting and long training times [25].

Over-fitting is defined as a neural network performing exceedingly well on training data but does not generalise to unseen data [25]. In the context of cardiac arrhythmia, the consequences can be life-threatening, as an over-fitting network would not be able to confidently classify new data. To counter this, one can employ dropout layers (of rate r, $0 \leq r \leq 1$) in their network [25]. A dropout layer will randomly disconnect rL neurons in any given layer, where L is the total number of neurons in a layer. This effectively simulates ensembling during training, reducing the overall risk that the network memorises the underlying training data [25].

To combat long training times one can make use of batch normalisation, which acts to normalise the input to each layer in a network [25]. Using this has been shown to speed up training times, it also allows the use of larger learning rates which further reduces the overall training time [25]. A secondary consequence of batch normalisation is that it can render dropout unnecessary due to its inherent regularisation effects [25].

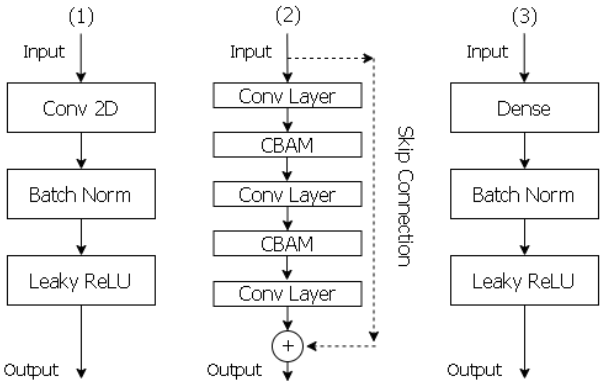

FIG. 10. A depiction of the architecture of the convolutional layers (1), the convolutional blocks (2), and the dense layers (3) used in the model for this study. The convolutional layers in (2) are those seen in (1). Also note that the CBAM and skip connection processes are optional, thus allowing for an ablation study of which performs best.

Seen in Figure 10 is the structure of the convolutional layers and blocks used. Figure 10.1 illustrates how convolutional layers were constructed, where batch normalisation is used before the non-linear activation. The choice of activation function was LeakyReLU. Using three of these layers the convolutional block in Figure 10.2 was produced. The components named 'CBAM' and 'Skip Connection' shall be explained in detail shortly. Finally, the dense layers also used in this study are detailed. Their structure is extremely similar to the convolutional layers used, where the outputs are normalised before being activated using LeakyReLU.

B. The convolutional block attention module

A convolutional block attention module (CBAM) is constructed via two layers: channel attention and spatial attention [26].

Channel attention squeezes the input feature (F) by pooling F into a global average and maximum simulta-

neously, this allows it to focus on where is important in F [4, 26]. This produces two new feature maps, F^c_{avg} and F^c_{max}, these are then independently passed to a multi-layer processing unit (MLP) to produce the channel attention map [4, 26]. The MLP was developed as per [4, 26], such that it consists of one hidden layer of units C/r, where C is the filter value of the previous convolutional layer and r a pre-determined reduction ratio [4, 26]. The channel maps produced by the MLP are then summed and passed through a sigmoid activation (σ) [4, 26]. This process is neatly illustrated via,

$$\boldsymbol{M_c}(F) = \sigma(\boldsymbol{W_1}(\boldsymbol{W_0}(F^c_{avg})) + \boldsymbol{W_1}(\boldsymbol{W_0}(F^c_{max}))), \quad (10)$$

here $\boldsymbol{W_1}$ and $\boldsymbol{W_0}$ represent the application of the MLP unit and all other symbols are as previously defined [4, 26].

As channel attention focuses on what is important, spatial attention naturally focuses on where is important [26]. To do this the input feature map is passed through a max and average pool along the channel axes which are then concatenated [26]. This concatenated product is then passed to a 2D convolutional layer which is activated using the sigmoid function [26]. Spatial attention can be expressed via,

$$\boldsymbol{M_s}(F) = \sigma(f^{k*k}([F^s_{avg}; F^s_{max}])), \quad (11)$$

where k represents the kernel size of the convolutional layer f. This study used a kernel of $k = 7$ as [26] found that performed best.

The total CBAM module was constructed as

$$\boldsymbol{M_{CBAM}}(F) = \boldsymbol{M_s}(\boldsymbol{M_c}(F)), \quad (12)$$

this was one of two variants that were accessible via [26]. The other possible variant was

$$\boldsymbol{M_{CBAM}}(F) = \boldsymbol{M_s}(F') \otimes F', \quad (13)$$

where $F' = \boldsymbol{M_c}(F) \otimes F$ [26]. It was found that the second instantiation of the CBAM module reduced the overall accuracy on validation and testing data to approximately 14% - a near guess.

C. Skip connections

Training a neural network often has a singular goal, to minimise the overall loss. This idea of minimisation often is presented in the form of analogy, such as traversing down a hill/valley. The concept is that the neural network will take steps of a pre-determined size (the algorithm's learning rate) with the aim of reaching the lowest point of the valley. However, these valleys, or loss surfaces, can be extremely complex and one may find themselves stuck in sub-optimal local minima. The level of complexity of loss surfaces can be seen in Figure 11(a), this is an example of a loss surface where [27] produced this for the case of ResNet-56. Therefore, if the loss surface is smoother and contains very few minima the network may reach an optimal solution much easier. This is the sole purpose of skip connections, their use can lead to the smoothing of the loss surface as seen in Figure 11(b).

The method of producing a skip connection can be seen in Figure 10.2. Here, the input to the convolutional block is taken and then summed to the output of the final layer in the same block. This is one method of instantiating a skip connection. Some alternatives could be using concatenation instead of a sum, using the skip connection between layers rather than over the whole block, or using a skip connection to step over one or more convolution blocks. Overall each method is valid and the choice of which one to use is case-dependent. However, they all should produce a smoothed loss surface which theoretically should be simpler for a CNN to traverse.

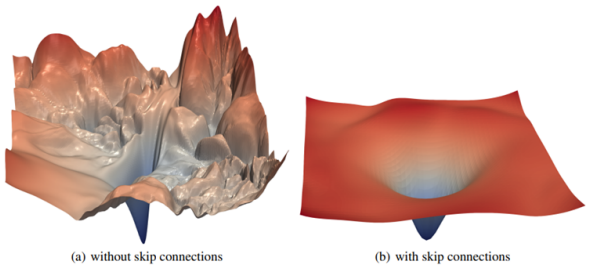

FIG. 11. An example loss surface of ResNet-56 before and after making use of skip connections [27]. Produced by Hao, et al. [27].

V. SINGLE TO MULTI-LEAD ANALYSIS

A. Evaluating a neural networks performance

Before discussing how the models were developed, a method of evaluating their performance must first be established. To do this several different metrics were calculated. First, the F1 score was calculated via the mean of the F_{ij} scores, those for the individual classes which are defined as

$$F_{ij} = \frac{2N_{ii}}{\sum_{j=1}^{9} N_{ij} + N_{ji}} \ [2,\ 4]. \quad (14)$$

This method has been retained from the previous semester [4]. Whilst the calculation of the F_1 scores allows for direct comparison with the CPSC participants, it is restrictive to only have a single evaluation metric when looking at a broader range of studies. To combat this several others were calculated.

One such metric is the accuracy score of the model, which is calculated via

$$accuracy(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n-1} \mathbf{1}(\hat{y}_i, y_i), \quad (15)$$

where n represents the total number of samples, i is the class number, and finally y and $\hat{y}$ the true and predicted labels respectively [28]. $\mathbf{1}(\hat{y}_i, y_i)$ is known as the indicator function and is defined as

$$\mathbf{1}(\hat{y}_i, y_i) := \begin{cases} 1 & \text{if } \ y_i = \hat{y}_i \ , \\ 0 & \text{if } \ y_i \neq \hat{y}_i \ . \end{cases} \text{[28]} \tag{16}$$

Note that the true definition of the indicator function is slightly more complex but in this specific case, the above is sufficient. Furthermore, the precision, sensitivity, and specificity of the network were calculated. They all depend on the TP (true positive), TN (true negative), FP (false positive), and FN (false negative) values [28]. These values were extracted via the multi-label confusion matrix from SKlearn [28]. These metrics are defined via

$$precision = \frac{TP}{TP + FP} \text{[28]}, \tag{17}$$

$$sensitivity = \frac{TP}{TP + FN}{}^{3} \text{[28]}, \tag{18}$$

$$specificity = \frac{TN}{TN + FP} \text{[28]}. \tag{19}$$

Note that for the specificity and sensitivity, these equations will produce a value for each class (nine in total), therefore, the mean of each was taken to provide a singular value.

B. Model development - an ablation study

Similar to [26] an ablation study was performed here in order to determine the best possible combination of model components; refer back to Section IV for details on each component. In pursuing an appropriate model, a simple CNN was first used, this had no skip connections, channel attention, or spatial attention modules were included.

In developing the model, only a single convolutional and fully connected layer were used initially. Naturally, this produced relatively poor results, therefore, the model complexity was expanded. This architecture expansion had the singular goal of achieving a validation accuracy greater than 75%. The expansion of the network resulted in more convolutional and pooling layers, however, this could only produce mediocre results ($\approx$ 60% validation accuracy). Therefore, recurrent style layers were trailed, these did not improve the model's accuracy or loss. As this failed the next step taken was hyper-parameter tuning, which produced the model seen in Table II. The

[3] Sensitivity is sometimes referred to as recall in literature.

Layer Type	Kernel	Kernel Size	Ouput Shape
Input and Rescale	-	-	128, 128, 1
Conv Block	[32]*3	3,3	128, 128, 32
Pooling	-	2,2	64, 64, 32
Conv Block	[32]*3	3,3	64, 64, 32
Pooling	-	2,2	32, 32, 32
Conv Block	[128]*3	3,3	32, 32, 128
Pooling	-	2,2	16, 16, 128
Conv Block	[256]*3	3,3	16, 16, 256
Pooling	-	2,2	8, 8, 256
Flatten	-	-	16,384
Dense Layer	128	-	16,384
FC Layer	9	-	9

TABLE II. The final CNN architecture used for the training of the individual models. The convolutional blocks are constructed as seen in Figure 10.2, also note that [32]*3 indicates all three convolutional layers in the block had a kernel of 32. The fully connected (FC) layer differs from the dense layer as its output is not normalised and is activated using the sigmoid function (as is the convention for a multi-label problem.

CNN produced a validation accuracy of 82%, this performance was checked by using it with the test set. As mentioned in Section II, this has been reserved and is completely independent of both the training and validation sets. Therefore, temporal leakage is impossible and the results achieved on the test set can be considered the true performance of the model. The model's performance can be seen in Table III, where the table uses only a subset of the metrics seen in section V A for simpler comparison.

Model	F1 Score	Accuracy	Precision
Semester 1	0.551	-	-
CNN	**0.9031**	0.8992	0.9468
CNN+Channel	0.8816	0.8936	0.9409
CNN+CBAM	0.8948	0.9047	0.9516
CNN+CBAM+SC	0.8963	**0.9068**	**0.9534**

TABLE III. The main model types tested and their resulting scores. Recorded here is the F1 score, accuracy, and precision. The best values are highlighted in bold. Note these models were trained on lead 7 data under-sampled to 12,000 data points. The F1 score of the model used in semester one is included for comparative purposes. The accuracy and precision were not calculated last semester and are thus not included [4]).

After achieving such high results, it was now time to attempt to beat them. To do this the CBAM module was implemented in two ways. The first only used the channel attention component (similar to last semester [4]), and the second made use of both channel and spatial attention. As seen in Table III, the full CBAM module outperformed both the CNN and the CNN+Channel

models on two of the three metrics used. Therefore, the CNN+CBAM model was taken as the best model and testing continued.

In an attempt to further these results skip connections as detailed in Section IV C were implemented. These further improved the already very good results, therefore, the final model is as seen in Table II, where the convolutional blocks can be seen in Figure 10.2.

The ablation study and all its associated results were achieved using lead 7 data, which was under-sampled to 12000 data points. This was done for several reasons: time efficiency, GPU memory, and lead 7 was the best performer when using the Mexican Hat wavelet last semester [4].

C. Stacked ensemble modelling using meta-learners

Using the model seen in Table II, 12 different models were trained independently, one for each ECG lead. Each model was trained for 75 epochs[4] making use of mini-batch stochastic gradient descent with a batch size of 32 and a variable learning rate. From these individual models, scores and accuracies can be extracted - see Appendix A. In the regime of ensemble learning these models constitute a level zero space Θ_0 of classifiers [29]. It is worth noting however that the 2D CNN failed to classify any arrhythmia meaningfully on leads V2, V4, and V6. This arose late in the training process, therefore, it has not been investigated to a satisfactory level. What comments can be made is that the error seems to arise from the CWT function used. Some potential possibilities are that the CWT is not well suited to these leads, the leads contain significant amounts of corrupted data, or the CNN is not complex enough to classify these. The latter two arguments are unlikely, however, as [9] made use of all 12 leads with an ensemble R-CNN and this study's CNN was more than complex enough to work on all other leads. However, excluding these 3 models from the ensemble leaves one task, to construct a level one space, Θ_1 [29]. Seen in Figure 12 is an example of an ensemble model, where the meta learner produces Θ_1. In order to make use of a meta-learner, first the predictions must be established. Therefore, each of the 12 models constructed prior was loaded and made predictions on the entire training set (no under-sampling was used) as well as the test set. These predictions were taken independently to avoid leakage between the two and produced two arrays of size (119085, 9, 12) and (5256, 9, 12), i.e. in the format (samples, classes, leads). Using this data a secondary ablation study was performed in order to determine the best meta-learner.

Several different meta-models were trailed, the first being linear regression. A linear regression algorithm can

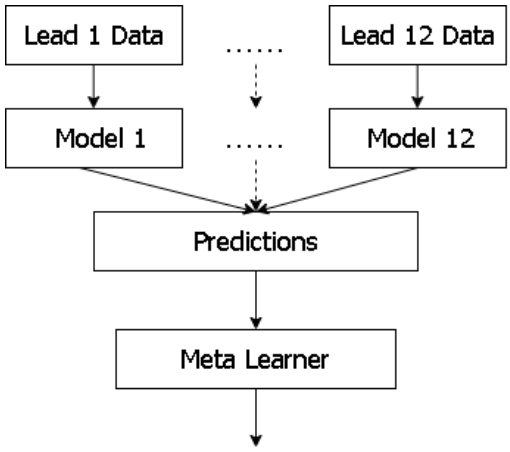

FIG. 12. An example of the ensemble structure that has been implemented here. Lead 1 through 12 blocks represent the individual models trained (Section V B). The meta learner block is abstract as several shall be trailed. Diagram inspiration: [30].

often be conveniently represented in terms of several matrices,

$$\boldsymbol{y} = \boldsymbol{\beta X} + \boldsymbol{\epsilon}, \tag{20}$$

where $\boldsymbol{X} = (x_{ij})_{m\times n}$ is the predictor matrix defined for targets $\boldsymbol{y} = (y_i)_{n\times 1}$ [31]. This format is analogous to the familiar equation for a line of best fit $y = mx + c$. Therefore, a solution may be found via a method of least squares, this is what the SKLearn library makes use of. The least squares method may be written as

$$Q(\beta) = (\boldsymbol{y} - \boldsymbol{\beta X})^T(\boldsymbol{y} - \boldsymbol{\beta X})^5[31]. \tag{21}$$

This method acts to minimise the distance magnitude between predictions and targets [31]. The next algorithm used was the K-nearest neighbour algorithm (KNN), also taken from the SKLearn library. This algorithm is based on the principle that target data can be divided into K groups, if a new target x' is then considered, whichever of the K groups x' is closest to will be of the same target label [32]. In order for this to work a distance measure must be employed, often it is the Minkowski metric [32]:

$$(ds)^p = (\sum_{i=0}^{q} dx_i^p)^{\frac{1}{p}}. \tag{22}$$

This establishes the distance measure $(ds)^p = |x' - x_j|^p$ in $\mathbb{R}^q$ space as the sum over the change in space between two targets, $dx_i = |(x_i)' - (x_i)_j|$ [32]. p defines the dimensions of the data space considered, $p = 2$ implies

[4] Early stopping was implemented to prevent major over-fitting, so the number of epochs is only approximate

[5] T indicates the matrix transpose

Euclidean space [32]. This information may then be used in a secondary calculation of

$$f(\boldsymbol{x'}) = \text{argmax}[\sum_{i \in N_K(\boldsymbol{x'})} \mathbf{1}(\hat{y}_i, y)], \tag{23}$$

where $f(\boldsymbol{x'})$ is the KNN classifier and $\mathbf{1}$ is the indicator function as defined in Section V A. The final algorithm taken from the SKLearn library was the decision tree classifier. These types of models comprise of several different sequential logical tests, each test performs a comparison between a numeric value (label) against a set of possible values (targets) [33]. The simplest of these tests would return a boolean value and then progress onward in one of two directions based on whether this boolean was true or false [33]. These directions being the prediction or further tests.

Layer	Kernel	Kernel Size	Activation
Conv 1D	64	3	ReLU
MaxPool 1D	(2,2)	-	-
Flatten	-	-	-
Dense	128	-	ReLU
Dense	64	-	ReLU
Dense	32		ReLU
Fully Connected	9		Sigmoid

TABLE IV. A depiction of the best-performing meta-model, the 1D CNN. Note that removing the convolution and max pool layer will give the structure of the ANN used in the study as well.

In order to make use of these SKLearn facilities the data used had to be reshaped into the format (samples $\times$ leads, 9). This was a predetermined requirement of the classifiers, therefore, there was no control over this. It is worthy of note the linear regression and KNN algorithms trained almost instantly, whereas the decision tree took significantly longer.

As previously mentioned some custom models were also used, these consisted of a simple ANN and 1D-CNN. These have both been previously described in Section IV. The structure of the ANN used for the meta-model may be seen in Table IV, where if the Conv1D and MaxPool 1D layers are removed the structure of the ANN can be seen. As this ANN produced promising results it was decided to further develop this into a 1D-CNN. A 1D-CNN was chosen over a 2D-CNN due to the data shape. Both models were trained for 100 epochs using SGD with a variable learning rate.

Table V illustrates the various results obtained for each of the algorithms considered. All models used obtained extremely similar results with the exception of the decision tree which under-performed. This under-performance can likely be attributed to decision trees not being well suited to the task. The 1D-CNN however, outperformed all other algorithms by at least .03%. Overall if one's aim is to acquire the fastest classification possible

Model	F1	Accuracy	Precision	Sensitivity	Specificity
Inidivdual Lead Models					
Lead 7	0.9077	0.9113	0.9577	0.9146	0.9946
Lead 1	0.8775	0.8790	0.9278	0.8912	0.9906
Ensemble Level 1 Models					
LR	0.9725	0.9511	1.0	0.9465	1.0
KNN	0.9743	0.9511	1.0	0.9500	1.0
DT	0.8374	0.8508	0.8301	0.8447	0.9864
ANN	0.9737	0.9511	1.0	0.9487	1.0
1D CNN	**0.9776**	0.9511	1.0	**0.9776**	1.0

TABLE V. The best (lead 7) and worst (lead 1) results for the individual models, note that the 'worst' model is only from those included in the ensemble - see Table VII in Appendix A. Following these are the results for various meta-models, the first three are Linear Regression (LR), K-Nearest Neighbours (KNN), and Decision Tree (DT) algorithms. These were taken from the SKLearn machine library. Following these are an ANN and a 1D CNN, these were custom-built for this study. The best overall metrics are emboldened.

whilst still maintaining a reasonable degree of accuracy the KNN algorithm would be recommended. This is due to the training times being near instantaneous. However, if the aim is purely the best possible results then naturally the 1D-CNN would be recommended. When testing this CNN the binary confusion matrices seen in Figure 13 were produced. From these matrices, it can be seen that

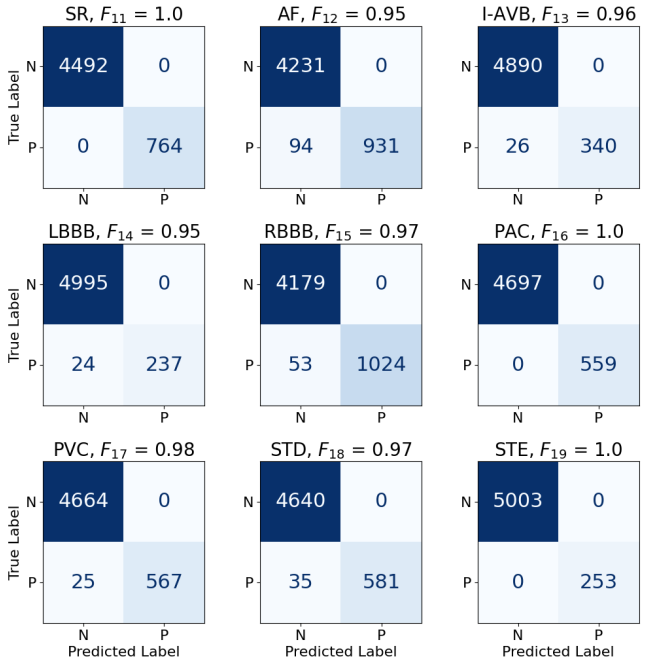

FIG. 13. The binary confusion matrices produced by the 1D CNN meta-model on the test data. Each confusion matrix represents a single class which is denoted in their titles, noted in the title is each leads F_{1i} score.

the ensemble model used produces perfect classification on classes SR, PAC, and STE. However, the other classes

whilst not perfect all produce a score of $F_{1i} \geq 0.95$.

VI. RELATED WORK

Seen in Table VI is a list of various studies encountered during the research phase of this project. All of these studies look at the classification of cardiac arrhythmia via various means. Included first in this list is this study's best-performing model (the 1D CNN ensemble), followed by the CPSC-2018 competition winner. After this, there is no particular order to the studies.

Out of all these studies, the one most reminiscent of the techniques used here is [36]. In their study, they took the ECG data and processed it using a discrete wavelet transform, segmented each recording to $2.5s$, made use of a GASF transform, and then classified the data using a CBAM-ResNet network [36]. The point of interest here is the GASF transform. The GASF intends to map the 1D ECG data in Cartesian coordinates to a polar coordinate system using trigonometric functions to transform it into a 2D matrix [36]. The polar coordinate encoding method used in [36] is expressed as

$$\begin{aligned} \phi_i &= arccos(x_i), \\ r_i &= i/N, \end{aligned} \tag{24}$$

here, $0 \leq x_i \leq 1$ and $x_i \in X$, where X is the normalised time series data of the ECG [36]. As well i is the timestamp where $i \in N$, N is the total time length of the ECG recording, in this case $N = 900$ [36]. Now the data is in polar coordinates, the GASF matrix is produced by the mapping of $\mathcal{G}_{ij} = cos(\phi_i + \phi_j)$, where $cos(\phi_i + \phi_j) := x_i \oplus x_j$ [36]. One issue throughout this study was the CWT is computationally expensive to perform, therefore, run times when converting the ECG data could be potentially very long. Hence, an alternative method (if computationally more efficient) is attractive. This could be the GASF transform, if this is used a comparison on time to transform should be made as well as on overall classification accuracy. From the work performed by [36] it seems reasonable that similar scores could be achieved, [36] found an F1 score of 0.8708 on the CPSC-2018 dataset. In their study, [36] did not train on the CPSC-2018 dataset and did not use an ensemble model [36]. Therefore, it appears a promising alternative to use the GASF transform.

The third entry in Table VI, [34], proposed a particularly interesting dataset above all else [34]. This dataset comprised of 27 classes and was constructed from four sub-datasets: CPSC-2018, CPSC-2018 EXTRA, PTB-XL, and Georgia [34]. The CPSC-2018 competition website does not refer to this 'EXTRA' database, therefore no comment can be made on the validity of this specific database [20]. Using these four sub-databases, [34] produced an extraordinarily large database consisting of 42,511 ECG recordings (with a total of 27 classes). This large dataset has its advantages and disadvantages. Naturally, it is desirable to have a singular framework that is capable of classifying many different types of arrhythmia, however, this would be extremely computationally demanding. [34] made use of ResNet50 currently one of the most advanced CNNs which is designed for image classification, this alone illustrates the complexity of classifying so many arrhythmias. However, [34] did achieve an accuracy of 0.9758, the best found amongst the literature surveys (including this study). This indicates that it could be beneficial in the future to alter the individual CNNs (the one seen in Table II) to either be more similar to ResNet or replace it entirely with ResNet50. Furthermore, it is evident that it is possible to accurately classify a large amount of arrhythmia. This extension to more arrhythmia is another avenue of study that is strongly encouraged, whether that be through combining several datasets or using another large-scale dataset.

VII. DISCUSSION AND FURTHER WORK

Throughout this study, techniques have been implemented to ensure the failings of last semester did not repeat [4]. These additional techniques led to the massive improvement seen across the individual models seen in Table III, where an F1 score of 0.8963 was found. The individual models were further expanded on by implementing an ensemble model containing 9 of the total 12 models. This achieved an F1 score of 0.9776 with the 1D CNN meta model. It is worth noting however, that whilst several different models were trailed they all achieved the same accuracy (except the decision tree). This could potentially be attributed to some unknown error, limitation of this specific model, or limitation of the pre-processing method.

Overall, the 1D CNN ensemble model has outperformed all of the surveyed literature on every metric, with the exception of [34] who outperformed this study by $\approx 2\%$ on overall accuracy. Furthermore, this study exceeded the winners of the CPSC-2018 challenge by $\approx 14\%$ on F1 score, a massive increase. The main issue with this method is that the CWT failed on leads V2, V4, and V6 leading to the CNN failing to classify any arrhythmia meaningfully on these leads. The underlying reason for this issue is currently unknown due to various time constraints. As this affected only a minority of the total data, excluding these leads from the ensemble did not significantly impact the overall results. It does mean however, that one-quarter of the total data is not in use. Going forward it should be a high priority to diagnose this issue, whether it lies in the data or the CWT method used and then rectify it. Unless the GASF transform works as intended and outperforms the CWT on run-time and evaluation metrics. The ensemble model's high-performance level is believed to be correlated to the use of the Pan-Tompkins algorithm for R-peak detection. In the previous semester, it was concluded that the primary issue was the segmentation method, where each ECG was reduced to $4s$ in length [4]. This is not an

Model	Author	Dataset	Classes	Indep. train/test	F1 Score	Accuracy	Precision	Sensitivity	Specificity
Ensemble CNN	Morris, J. Ormrod, J.	CPSC-2018	9	Yes	**0.9776**	0.9511	**1.0**	**0.9776**	**1.0**
Ensemble C-RNN	Chen T. et al. [9]	CPSC-2018 (comp winner)	9	Yes	0.8370	-	-	-	-
ResNet50	Sakli, N. et al. [34]	N/A	27	Yes	-	**0.9758**	0.8885	-	-
Inception RNN	Ni, J. et al. [35]	CPSC-2018	9	Yes	0.8970	-	-	-	-
CBAM ResNet	Ma. et al. [36]	MIT-BIH & CPSC-2018	5	Yes	0.8708	0.8695	0.8721	0.8695	0.9639

TABLE VI. An overall result comparison with various studies performed on many datasets. Listed first is this study followed by the CPSC-2018 competition winner, after that entries are in no particular order. The best scores are emboldened. Entry 3, Sakli, N. has a not applicable dataset as they used a combination of datasets, this combination included CPSC-2018 [34]. Entry 5, [36], has two datasets listed, they trained on the 80% of the MIT-BIH dataset and then tested on lead II of the CPSC-2018 dataset.

unpopular method as [36] segmented their data to $2.5s$. However, the use of the Pan-Tompkins algorithm in conjunction with the sliding window generated a consistency in the data. This consistency being that every signal passed to the CNN contained exactly 3 heartbeats, no matter the arrhythmia. Subsequently allowing the CNN to generalise more easily as the training data is more representative of the testing set.

While there is not a significant amount of room for improvement in terms of the overall classification metrics, several pre-processing methods could be trailed to determine their impact. For example, using discrete wavelet transforms to clean ECG data is an extremely common method, such as in [36]. Making use of this independently or in conjunction with the established bandpass filter could potentially lead to improvements. Instead of under-sampling during training, there is the potential to over-sample the data instead. Over-sampling is the method of boosting minority classes to be more in line with the number of entries in the majority classes [18]. This could be done in several ways, however, the one which appears most appealing is the use of generative adversarial networks (GAN). GANs comprise of a generator (generates artificial data) and a discriminator (checks if data is real or artificial), through these a GAN is capable of producing highly realistic data which could be used for training or testing a neural network [37]. Through the use of GANs the minority classes could be augmented providing more unique data samples for the CNN to learn from.

A potential source of error in this project arises from the nature of the data, it is labelled by people, cardiologists. This is one potential downfall of using a supervised learning algorithm (like this study), there is always the possibility of human error. As the goal of these machine learning systems is to aid cardiologists it is likely best that they are developed independently of them. Therefore, it is highly recommended that an unsupervised approach should be taken in the future.

VIII. CONCLUSION

The CPSC-2018 database contains 6,877 patients, each of which possesses 12 leads of ECG data of varying time length, t ($6 \leq t \leq 60$). In processing the database it was found that several patient recordings either contained zero signal (on one or more leads) or exceeded the $60s$ time limit specified by the CPSC website [2]. This led to the removal of these patients from the study, resulting in a total of 5,738 patients being used.

Using these patients, the ECGs were processed such that they could be used in a neural network. The ECG signals were filtered using a Butterworth bandpass filter of passband $[1, 50]$Hz, where the filter order was $n = 1$. The processed data was then normalised and passed to the Pan-Tompkins algorithm for R-peak detection. Using this information a sliding window algorithm segmented each recording into N 3-beat segments. Finally, the ECGs were passed through a CWT in order to convert to time-frequency space.

The CWT data was resized to 128x128 black and white images such that it was appropriate for use in a 2D CBAM-CNN. Twelve of these CBAM-CNN models were trained, one for each lead, these constructed the level-0 space of a stacked ensemble model. The CBAM-CNN was unable to train on leads V2, V4, and V6 resulting in these leads being excluded from the ensemble model. Various different meta-models (level-1 ensemble space) were tested, the best being a lightweight 1D CNN. The CNN was able to correctly classify the predictions passed to it resulting in a F1 score of 0.9776 on the testing data used. This is a considerable improvement upon last semester's work and it outperforms all surveyed literature in this study (Table VI). This is with the exception of the accuracy metric, where this study performs second best. This significant improvement is believed to be directly correlated to the use of the Pan-Tompkins algorithm and the sliding window. These allowed for the production of uniform images, each containing 3 beats per segment. This is supported by [36] as they segmented each signal

to 2.5s windows and achieved significantly worse results than this paper.

Overall, this study can be considered a success, however, in the future, the CWT method should be heavily investigated when regarding its performance on leads V2, V4, and V6.

[1] R Gonzalez, L Romero, J Gomis-Tena, B Trenor, JM Ferrero, and J Saiz. Combined effects of acquired lqt syndrome by dofetilide and reduced repolarization reserve on human ventricular action potential: A simulation study. In *2009 36th Annual Computers in Cardiology Conference (CinC)*, pages 445–448, 2009.

[2] Feifei Liu, Chengyu Liu, Lina Zhao, Xiangyu Zhang, Xiaoling Wu, Xiaoyan Xu, Yulin Liu, Caiyun Ma, Shoushui Wei, Zhiqiang He, Jianqing Li, and Eddie Ng. An open access database for evaluating the algorithms of electrocardiogram rhythm and morphology abnormality detection. *Journal of Medical Imaging and Health Informatics*, 8:1368–1373, 09 2018.

[3] KT Macleod, SB Marston, PA Poole-Wilson, NJ Severs, and PH Sugden. Cardiac myocytes and the cardiac action potential. *Oxford Textbook of Medicine, fifth ed. Oxford University Press, Oxford*, 2010.

[4] Morris Jordan. Convolutional neural networks as a classifier for cardiac arrhythmias. 12 2022.

[5] Sharon Ann George, Zexu Lin, and Igor R. Efimov. *Basic Principles of Cardiac Electrophysiology*, pages 3–32. Springer International Publishing, Cham, 2020.

[6] Flavio H Fenton, Elizabeth M Cherry, and Leon Glass. Cardiac arrhythmia. *Scholarpedia*, 3(7):1665, 2008.

[7] Jehoshaph Chandran. Classification of electrolytic imbalances using electrocardiography. 10 2014.

[8] Selcan Kaplan Berkaya, Alper Kursat Uysal, Efnan Sora Gunal, Semih Ergin, Serkan Gunal, and M. Bilginer Gulmezoglu. A survey on ecg analysis. *Biomedical Signal Processing and Control*, 43:216–235, 2018.

[9] Tsai-Min Chen, Chih-Han Huang, Edward S.C. Shih, Yu-Feng Hu, and Ming-Jing Hwang. Detection and classification of cardiac arrhythmias by a challenge-best deep learning neural network model. *iScience*, 23(3):100886, 2020.

[10] Qin Qin, Jianqing Li, Li Zhang, Yinggao Yue, and Chengyu Liu. Combining low-dimensional wavelet features and support vector machine for arrhythmia beat classification. *Scientific Reports*, 7(1), 2017. Cited by: 59; All Open Access, Gold Open Access, Green Open Access.

[11] Sonal K. Jagtap and M. D. Uplane. The impact of digital filtering to ecg analysis: Butterworth filter application. In *2012 International Conference on Communication, Information Computing Technology (ICCICT)*, pages 1–6, 2012.

[12] AA Fedotov. Selection of parameters of bandpass filtering of the ecg signal for heart rhythm monitoring systems. *Biomedical Engineering*, 50:114–118, 2016.

[13] Belle A Shenoi. *Introduction to digital signal processing and filter design*. John Wiley & Sons, 2005.

[14] Mokhtar Shouran and Elmazeg Elgamli. Design and implementation of butterworth filter. *International Journal of Innovative Research in Science, Engineering and Technology*, 9(9), 2020.

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

[16] Shital L Pingale. Using pan tompkin 's method, ecg signal processing and diagnose various diseases in matlab. In *Proceedings of IRF International Conference*, pages 57–61, 2014.

[17] Jyoti Bali, Anilkumar Nandi, PS Hiremath, and Poornima G Patil. Detection of sleep apnea in ecg signal using pan-tompkins algorithm and ann classifiers. *Compusoft*, 7(11):2852–2861, 2018.

[18] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248, 2020.

[19] Tao Wang, Changhua Lu, Yining Sun, Mei Yang, Chun Liu, and Chunsheng Ou. Automatic ecg classification using continuous wavelet transform and convolutional neural network. *Entropy*, 23(1), 2021.

[20] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O8217;Leary. Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237, 2019.

[21] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, 1996.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[24] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.

[25] Christian Garbin, Xingquan Zhu, and Oge Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, 79:12777–12815, 2020.

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

[27] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018.

[28] 3.3. metrics and scoring: quantifying the quality of predictions — scikit-learn 1.2.2 documentation. `https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score`. (Accessed on 04/18/2023).

[29] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[30] Alok Kumar and J Mayank. Ensemble learning for ai developers. *BApress: Berkeley, CA, USA*, 2020.

[31] Xiaogang Su, Xin Yan, and Chih-Ling Tsai. Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(3):275–294, 2012.

[32] Oliver Kramer and Oliver Kramer. K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23, 2013.

[33] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.

[34] Nizar Sakli, Haifa Ghabri, Ben Othman Soufiene, Faris Almalki, Hedi Sakli, Obaid Ali, Mustapha Najjari, et al. Resnet-50 for 12-lead electrocardiogram automated diagnosis. *Computational Intelligence and Neuroscience*, 2022, 2022.

[35] Jian Ni, Yingtao Jiang, Shengjie Zhai, Yihan Chen, Sijia Li, Amei Amei, Dieu-My T Tran, Lijie Zhai, and Yu Kuang. Multi-class cardiovascular disease detection and classification from 12-lead ecg signals using an inception residual network. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1532–1537. IEEE, 2021.

[36] Ke Ma, Chang'an A. Zhan, and Feng Yang. Multiclassification of arrhythmias using resnet with cbam on cwgan-gp augmented ecg gramian angular summation field. *Biomedical Signal Processing and Control*, 77:103684, 2022.

[37] Esteban Piacentino, Alvaro Guarner, and Cecilio Angulo. Generating synthetic ecgs using gans for anonymizing healthcare data. *Electronics*, 10(4):389, 2021.

Appendix A: Individual Model Results

Lead	F_1	F_{AF}	F_{Block}	F_{PC}	F_{ST}	Accuracy	Precision	Sensitivity	Specificity
I	0.8775	0.9425	0.8692	0.8257	0.8866	0.8790	0.9278	0.8912	0.9906
II	0.8766	0.9213	0.8744	0.8305	0.9020	0.8963	0.9442	0.9083	0.9926
III	0.8948	0.9247	0.8972	0.8596	0.9032	0.8908	0.9382	0.9066	0.9920
AVR	**0.9197**	**0.9438**	0.9209	**0.8909**	**0.9388**	0.8965	0.9439	0.9053	0.9927
AVL	0.8726	0.8706	0.8890	0.8595	0.8298	0.8923	0.9407	0.9052	0.9923
AVF	0.8719	0.8889	0.8469	0.8571	0.8868	0.8921	0.9394	0.9073	0.9921
V1	0.9077	0.8889	**0.9302**	**0.8909**	0.8541	**0.9113**	**0.9577**	**0.9146**	**0.9946**
V2	0.0113	0.0	0.0	0.0	0.0920	0.0432	0.0022	0.8889	0.1111
V3	0.8809	0.9302	0.8641	0.8547	0.8800	0.8910	0.9388	0.9030	0.9920
V4	0.0302	0.2717	0.0	0.0	0.0	0.1771	0.0363	0.8889	0.1111
V5	0.8787	0.8471	0.8529	0.8468	0.9216	0.8817	0.9294	0.8984	0.9908
V6	0.0238	0.0	0.0	0.0	0.2051	0.1003	0.0131	0.8889	0.1111

TABLE VII. All values of significance are stored in this table, the best values for each metric are emboldened. It is between lead 4 and 7 for the position of the best model. Lead 7 however, performs better on more metrics and therefore can be considered the best lead, agreeing with last semester. Whereas, the worst lead, that was included in the ensemble is evidently lead 1. The CNN however, failed to train on leads 8, 10, and 12 hence the terrible results. These three leads for this reason were not included in the ensemble.