Classification of ECG signals using a 1-D residual Convolutional Neural Network

Jeremias Ferrao (S4626451)
Mirko Borszukovszki (S4745000)
Stanislav Ivanov (S4739523)
Levente Kis (S4765389)

Department of Artificial Intelligence
University of Groningen

October 5, 2023

Abstract

The human heart is a biological marvel where failure to function properly can lead to catastrophic consequences. Therefore, there is a great incentive for the rapid and accurate diagnosis of heart disease. An electrocardiogram (ECG) is a test that measures the electrical signals of the heart. However, interpreting ECGs is a complex process that requires years of professional training and experience. This project makes use of a one-dimensional residual convolutional neural network (ResNet) to classify individual heartbeats using the PhysioNet MIT-BIH Arrhythmia database. Our pre-processing pipeline makes use of the Pam-Tompkins algorithm to extract features and the SMOTE oversampling technique to over-sample the minority heartbeats. We make use of 5-fold cross-validation to determine the optimum regularization hyper-parameter and assess our model using accuracy, precision, sensitivity, and specificity. An analysis of our model with these metrics reveals that it is comparable with other ECG classification techniques. However, there is scope for improvement as our model still struggles to properly classify minority heartbeat classes in the database.

Contents

1 INTRODUCTION

The human heart stands as an unwavering sentinel in the realm of physiological fortitude. It is a ceaseless guardian which tirelessly fulfils its noble duty of bestowing essential nutrients and energy upon every cell of the body. Thus, the failure of this vital organ for even a moment could lead to dreadful consequences. Additionally, cardiovascular failure is one of the leading causes of death in the world with a staggering 18 million fatalities annually (World Health Organization [WHO], 2023). Taking this into consideration, there is a great incentive for the rapid diagnosis and treatment of heart disease.

An electrocardiogram (ECG) serves as an informative description of the heart which depicts its electrical activity over a period of time. This signal can be procured easily through the placement of sensors over a subject's chest. However, the interpretation of ECGs is a complex task that requires years of demanding training and experience. As a consequence, individuals with the required expertise are uncommon and overloaded with work.

In order to ease the burden of cardiologists, complex machine learning (ML) models have been created to classify pathological heartbeats from ECG signals. Likewise, the aim of this project is to create a model that can accurately analyze ECGs and aid in the early detection of cardiovascular conditions. By leveraging the power of deep learning algorithms, this model should be capable of analyzing the intricate patterns and waveforms present in ECG signals and identifying the corresponding class of the heartbeat. This model will be trained on the data of several individuals without fine-tuning to ensure generality.

We were particularly inspired by Khan et al. (2023)'s usage of a one-dimensional residual convolutional neural network (ResNet) which attained an overall accuracy of 98.63% in classifying ECG signals. However, their model does not make use of regularization to prevent overfitting. As a result, we will build upon the same architecture described in their paper, and modify the convolutional layers to make use of the appropriate regularization technique and hyper-parameter value. We will also make use of the MIT-BIT Arrhythmia Database (Moody & Mark, 2001) which is the same dataset used by Khan et al. (2023). This is an extensive collection of annotated ECG signals widely recognized and employed in the field of arrhythmia and ECG research.

Once our model is trained, it will be evaluated using the measures of accuracy, precision, sensitivity, and specificity to properly assess its ability to predict disease. To gain a better understanding of our model's strengths and weaknesses, we will also examine class-wise performance using the same metrics. We hope to achieve similar success as reported by Khan et al. (2023) since our architecture will be derived from their paper. We will also make our code available at `https://github.com/Jazhyc/ECG-Classification` to facilitate inspection and replication.

2 DATA

For this project, we utilized the popular MIT-BIH Arrhythmia Database. This rich dataset was created by the Massachusetts Institute of Technology (MIT) and Beth Israel Hospital (BIH) to support research in ECG analysis and arrhythmia detection. The database contains 48 half-hour excerpts[1] of ECG recordings which were obtained from 47 subjects by placing two electrodes on their chest. Therefore, each record contains two signals which indicate electrical activity at different regions of the heart (Figure 1). The sampling frequency of these signals was 360 Hertz and the amplitude, measured in milliVolts (mV), was spread over a $\pm$ 5 mV range.

This dataset is particularly useful for machine learning since it has been manually annotated by independent cardiologists. These annotations mention the class and location of the recorded heartbeats in the ECG data. In total, there are around 110,000 annotations and 23 classes of heartbeats. However, the American Association of Medical Instrumentation (AAMI, 1998-2008)[2] recommends that these 23 classes should be consolidated into 5 categories which indicate normal heartbeats and 4 types of arrhythmias. The mapping of the classes into the categories can be observed in Figure 2. The remaining classes which do not fall under a category can be removed from the dataset.

Upon examination of heartbeats from the ECG data, one might uncover what is known as a QRS complex. As described by Goldberger et al. (2018), this is the main spike or the most obvious part of the ECG. A complete QRS complex (which represents a normal heartbeat) includes the following features which occur in rapid succession: An initial negative voltage dip known as a Q wave, a large positive voltage increase known as an R wave, and a second negative voltage dip known as an S wave. However, not every QRS complex contains all three waves. The QRS

[1]Two of the records were obtained from the same person. Thus, the number of records is higher than the number of subjects

[2]We could not access this book but obtained the mapping of classes from Table 1 in Mousavi & Afghah (2018)

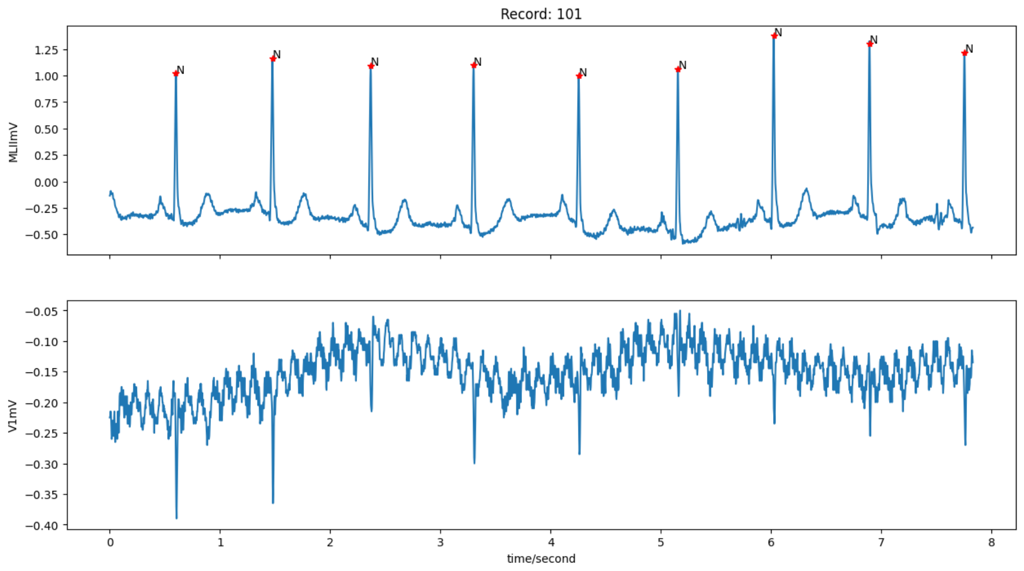

Figure 1: Slice of annotated ECG signal obtained from the dataset. The red stars denote the locations of R waves which represent large voltage increases observed due to ventricular depolarization. In this particular record, one sensor recorded the modified limb lead II (MLII) while the other measured the modified lead V1. To our knowledge, MLII and V1 represent different regions of the heart. Both of these sensors measured the amplitude in milliVolts

Category	**Class**
N	• Normal beat (N) • Left and right bundle branch block beats (L,R) • Atrial escape beat (e) • Nodal (junctional) escape beat (j)
S	• Atrial premature beat (A) • Aberrated atrial premature beat (a) • Nodal (junctional) premature beat (J) • Supraventricular premature beat (S)
V	• Premature ventricular contraction (V) • Ventricular escape beat (E)
F	• Fusion of ventricular and normal beat (F)
Q	• Paced beat (/) • Fusion of paced and normal beat (f) • Unclassifiable beat (U)

Figure 2: Categories of heartbeats in the MIT BIH dataset based on the AAMI standard. Reprinted from Mousavi & Afghah (2018)

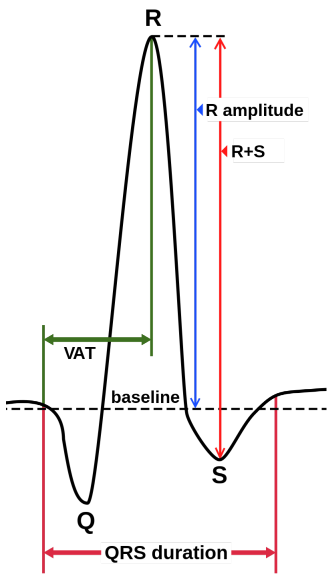

Figure 3: Schematic representation of a QRS complex by Häggström (2010). The Q, R and S waves are annotated along with other features of interest.

Table 1: Distribution of the heartbeat classes in the MIT BIH dataset as per AAMI standards. We count the number of occurrences based on their appearance in a channel. Since two channels are present in every record, the number of occurrences is near twice the number of annotations.

Category	Occurences	Percentage
N	181,178	82.8
Q	16,012	7.31
V	14,472	6.61
S	5,558	2.54
F	1,606	0.7
Total	218,826	100.00

complex is of interest since it represents the electrical impulse as it spreads through the ventricles and indicates ventricular depolarization or contraction. A number of important features can be discerned from the QRS complex as highlighted in Figure 3. The ideal goal of our model is to find these features and analyze their statistical relationship to the respective heartbeat class. Upon completion of this process, the model should gain the required understanding to classify heartbeats based on the ECG signal.

An important aspect that we must account for is the variance of ECG signals between records and sensors. For example, the upper and lower signal in Figure 1 demonstrates that the shape of the QRS complex can vary greatly for the same heartbeat class. We must also take the noise present in the signal into consideration due to several factors such as muscle movements and the usage of different sensors. An examination of the number of classes as seen in Table 1 reveals an additional problem in the form of class imbalance. Here, the number of occurrences of normal heartbeats far surpasses the combined count of the four arrhythmias. Addressing these issues upfront is crucial and will determine the ultimate success of the project.

3 Methods and experiments

This section describes the procedures used in our project. We will split the explanation of our machine-learning pipeline into 5 components which cover the following aspects: pre-processing of ECG data, balancing of minority heartbeat classes, model architecture, the configuration of hyperparameters and regularization, and model evaluation.

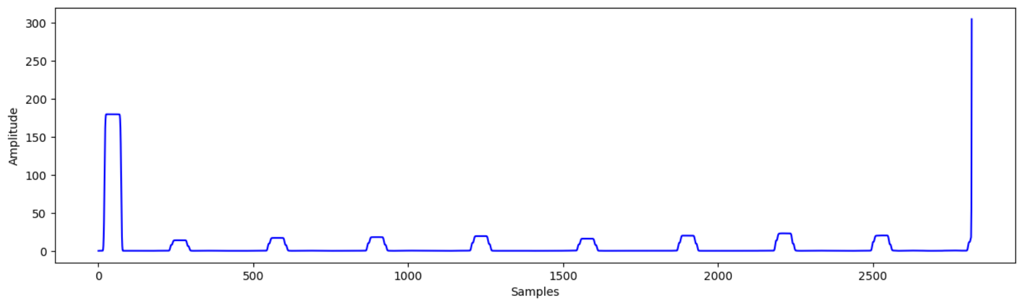

Figure 4: Slice of the resulting signal from applying the Pan Tompkins on the MLII signal present in Figure 1. A consequence of the moving target integration in the Pan Tompkins algorithm is that the first and last QRS complex in the entire signal is disproportionately sized.

3.1 Pre-Processing

In the analysis of electrocardiogram (ECG) data, pre-processing plays a crucial role to ensure the quality of the data, highlight key signal information, and simplify subsequent stages of analysis. This stage in our analysis pipeline involves several techniques to confront challenges such as noise, artefacts, and signal variation in ECG data due to the placement of sensors. To set up the ECG data for further examination, we adopt a sequence of pre-processing steps including signal enhancement and signal segmentation based on the annotations in the dataset.

3.1.1 Signal Enhancement

The first phase of our pre-processing applies the Pan-Tompkins algorithm to process the ECG signals (Pan & Tompkins, 1985). The algorithm applies a series of convolutions in succession to make QRS complexes more prominent and eliminate noise. At a high level, the algorithm studies feature such as amplitude, slope, and duration to accurately identify the QRS complexes. This enhancement step facilitates better visualization and subsequent analysis of cardiac activity. There are four main filters used in the algorithm which we will explain abstractly:

1. Bandpass Filter: The important features of a heartbeat lie in a 5-15 Hz range. Other variations in the ECG data that lie outside of this range can be attributed to the noise either due to the sensor or muscle movements. Thus, a low pass filter is applied to the raw ECG data with a frequency cutoff of 11 Hz followed by a high pass filter with a cutoff of 5 Hz. The resulting ECG signal by applying these two filters should be free from noise which will allow our model to focus on only the relevant features.
2. Derivative Filter: The second step is to differentiate the filtered signal to obtain information about the rate of change in voltage of the signal. This filter provides useful information regarding the slopes in the QRS complexes.
3. Squaring: Here, the differentiated signal is squared to amplify the features extracted from the previous step. In particular, this step enhances the presence of QRS complexes and makes them more prominent compared to other complexes in the ECG data.
4. Moving Window Integration: Finally, the squared signal is integrated using a moving window to obtain smoother waveforms. This step is useful since it provides information about the width of a QRS complex. The size of the moving window is generally 0.15 times the sampling frequency which is roughly the size of a QRS complex.

We used an implementation of the Pan-Tompkins algorithm by Sharma (2021) to avoid mistakes in implementation since we were not confident in our understanding of signal processing. An example of the processed signal obtained after using the algorithm is illustrated in Figure 4.

3.1.2 Signal Segmentation and Annotation

After applying the Pan Tompkins algorithm, we segment the processed ECG signals based on the corresponding annotation of R waves. Segmenting signals is crucial in pre-processing as it allows for the isolation of individual heartbeats or specific segments for more detailed analysis. More importantly, the segmentation of heartbeats into fixed slices of a signal is necessary since the number of input units in our model remains constant.

Using the QRS complexes as reference points, we need to determine the start and end positions of each heartbeat or segment in the ECG signal. To accomplish this process, we mark the start as 100 samples before the R wave and the end as 87 samples after. Therefore, each heartbeat is 188 samples long and serves as the input to our model. This

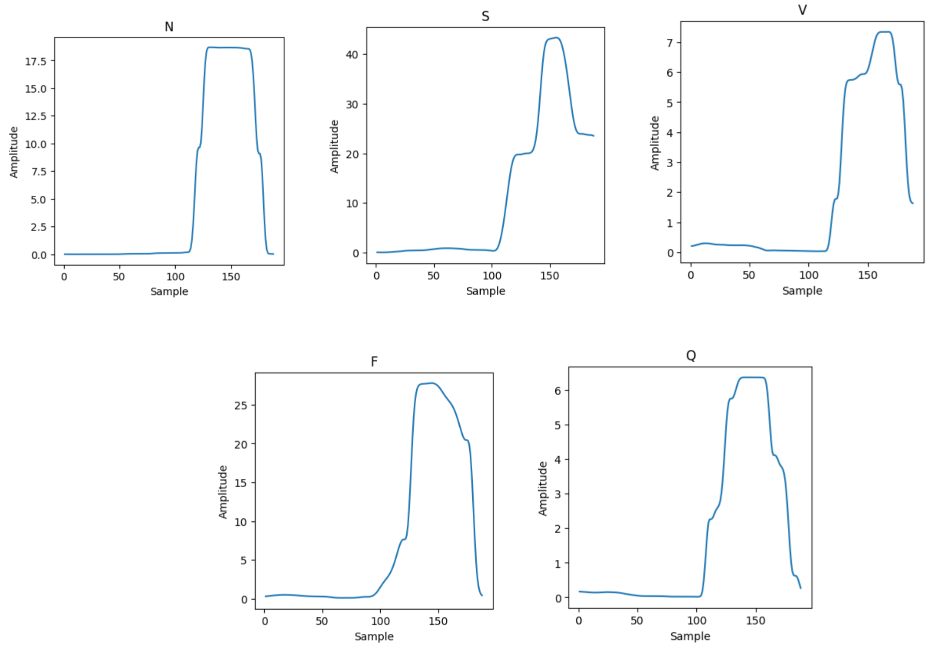

Figure 5: Example of a heartbeat from each class after pre-processing.

sample length of 188 appears to be the standard in the literature since it completely covers the smallest heartbeat class in the dataset (Khan et al., 2023; Rahman Khan et al., 2020). Figure 5 depicts an example segment from each class.

After the segmentation process, we can change the labels present in the MIT BIH dataset to follow the AAMI standard as depicted in Figure 2. Classes which do not fall in the categories are removed and the resulting count of all categories is described in Table 1. We also make sure to remove the first and last heartbeat due to their anomalous size as seen in Figure 4. This represents the removal of 96 samples from 200000+. As a result, no substantial data is lost and the model does not try to optimize its weights for these outliers.

3.1.3 One Hot Encoding and Data Splitting

The variable that we are trying to predict is the class of heartbeat from the ECG data. Since this variable is discrete, we will make use of one hot encoding to convert the discrete values into binary feature vectors. Thus, one hot encoding will convert the class variable into multiple binary variables, with one representing each class, and with a '1' for the true class and '0' for the others. After the encoding process, we create the randomized training and testing sets of data using an 80:20 ratio.

3.2 Data Balancing

As illustrated in Table 1, the N class accounts for nearly 80% of all heartbeats whereas the remaining 4 classes only account for 20%. If we were to train our model on this dataset, then there would be a considerable bias towards the majority class which leads to a high likelihood of misclassification of the other classes. To remedy this problem, we can duplicate the samples present in the minority classes until the distribution of classes is uniform. However, this duplication does not add any new information which the model can learn.

A technique known as the Synthetic Minority Oversampling Technique (SMOTE) addresses the class imbalance problem by generating synthetic data (Chawla et al., 2002). This represents a type of data augmentation which generates new data points that closely resemble the minority classes but vary in some aspects. As a result, the model is made more robust since the synthetic data points make it harder for overfitting to occur due to their differences from the original data points.

Formally, SMOTE makes use of the K-nearest neighbours technique which uses the Euclidean distance to determine the neighbours to a given data point. Once the nearest neighbours are determined, the synthetic data is generated using the below equation.

$$x^{syn} = x^i + (x^j - x^i)\delta$$

Here x^{syn} refers to the new synthetic data point, x^i is a selected data point from the minority class that is being oversampled, x^j is a randomly selected nearest neighbour of x^i in the same class, and δ is a factor whose value is randomized over a range of [0, 1] every time a data point is generated. Intuitively, SMOTE generates a new data point that lies somewhere between the minority class and its neighbour along a multi-dimensional space. The synthetic data points are generated until all classes are uniformly distributed.

We only make use of SMOTE during the training phase of the model by applying it to the training data. The testing data remains in its original distribution since we want to measure the performance of the model on data as it appears in reality.

3.3 Model Architecture

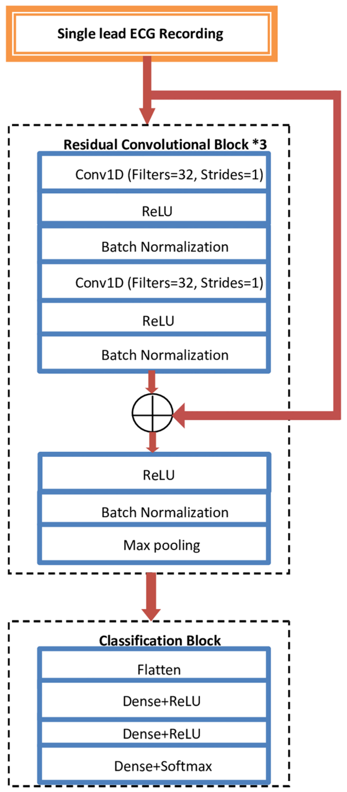

Figure 6: Architecture of the ResNet model used by Khan et al. (2023). Reprinted from their paper. Our model utilizes the same structure but regularization is added to the two convolutional layers in a residual block

The architecture of our model is derived from Khan et al. (2023) as observed in Figure 6. This is a residual neural network which takes an ECG segment as input and produces a label that matches one of the 5 classes in the dataset. A problem that troubles traditional neural networks is the loss of information as the number of layers increases due to vanishing or exploding gradients. Residual neural networks were created to address this issue and facilitate the creation of models with a greater number of layers (He et al., 2016). Residual networks accomplish this feat by refreshing processed data with the input after it has passed through several layers. The point where the data is refreshed is known as a skip connection which forms a residual block when combined with prior layers. Figure 7 illustrates this concept vividly. Mathematically, a skip connection can be expressed as:

$$y = f(x) + x$$

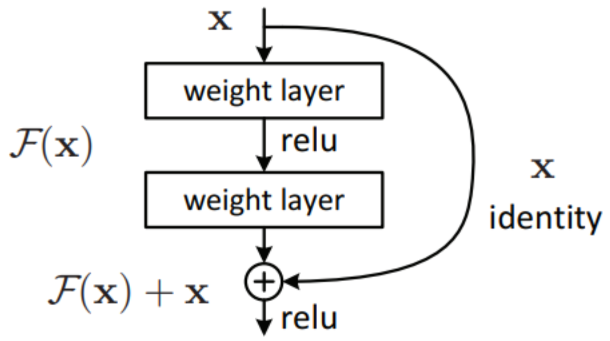

Figure 7: Residual Block in a deep residual network. The skip connection is denoted by the + sign

where y represents the output of a residual block, x represents the input, and f(x) is the processed input after it has been passed through the layers present in a residual block.

We will provide a high-level overview of the architecture and then further elaborate on each component in the proceeding sections. There are 3 residual blocks in our model which are used to extract important features from the ECG data. In each block, there are two convolutional layers which are fed into activation layers to enable the model to learn non-linear information. The output of these layers is then passed into a batch normalization layer to reduce the internal covariate shift and speed up the convergence of the data. A skip connection is then used to refresh the output data with input which is then passed into another activation and batch normalization layer. The last layer present in a residual block is the maximum pooling layer which reduces the dimensionality of the output. The resulting output from the residual blocks is passed into a classification block which flattens it into a single-dimensional vector. This is necessary since there are 3 fully connected layers which harvest information required for classification. The final layer uses a softmax activation function which provides the probabilities of class of heartbeat.

3.3.1 Input Layer

The starting point of any supervised machine learning model. The segmented ECG signals that we obtained will be fed to this layer. The inputs contain 188 features as numbers which represent a time series.

3.3.2 1D Convolutional Layer

The process of classification begins with Convolutional 1D layers (Conv1D). Unlike Convolutional 2D layers that process 2D data (like images), Conv1D layers are specifically optimized for one-dimensional sequential data such as ECG signals. These layers employ a series of sliding filters, also known as convolutional kernels, that pass over the input data. Each filter is designed to recognize a specific local pattern within the input sequence, and through the convolution operation, it produces a feature map that represents the presence of that pattern at different locations. The Conv1D layers learn to recognize multiple features simultaneously, capturing both short-term and long-term temporal dependencies inherent in the ECG signal. The convolutional layers will make use of 32 filters of size 3 and possess a stride of 1 which are the same values used by Khan et al. (2023). These values indicate that 32 feature maps are generated where each point in the feature map is determined by the 3 nearest values at the same point in the input. The stride of 1 indicates that the filters slide over each feature in the input sequentially without skipping any value.

In contrast to Khan et al. (2023), we will modify our convolutional layers to make use of L2 regularization. This involves adding a penalty of the form below to the optimization process used by the convolutional layer. We only add regularization to this layer since it serves as the key method for our model to extract information.

$$\lambda \sum_{i=1}^{n} W_i^2$$

Here, λ represents the strength of regularization, n is the number of weights in the layer, and W represents the value of the weight. Thus, L2 regularization adds a penalty which takes into account the squares of all the weights in the layer. This addition prevents the layer from taking on extreme weight values that create a perfect fit for the training data. As a result, overfitting is less likely to occur. For this purpose, we will determine the optimum value of λ through cross-validation.

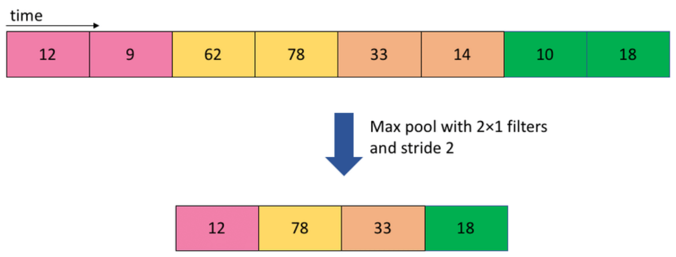

Figure 8: Example of a one-dimensional maximum pooling layer with a pooling size of 2 and stride of 2. Reprinted from Mohammadian Rad (2019)

3.3.3 Rectified Linear Unit Activation Layer

Activation functions introduce non-linearity to neural networks, allowing them to learn and model complex patterns. The Rectified Linear Unit (ReLU) activation function specifically transforms the input by setting negative values to zero and leaving positive values unchanged. Mathematically, ReLU can be defined as:

$$ReLU(x) = max(0, x)$$

where x represents the input. ReLU has become a popular choice for activation functions due to its simplicity and effectiveness. It helps the network capture and propagate positive activations while introducing non-linearity, enabling the network to learn more complex representations.

3.3.4 Batch Normalization Layer

Batch Normalization is another crucial component of our architecture. During the training of deep neural networks, the distribution of each layer's input can change as the weights of the previous layer change. This phenomenon, known as the internal covariate shift, can slow down training and make it harder for the network to converge. Batch Normalization mitigates this problem by normalizing the layer inputs. It calculates the mean and standard deviation of the input within each batch and adjusts them to have zero mean and unit variance. The normalized inputs are then scaled and shifted using learnable parameters (γ and β), ensuring that the network can still learn and represent different distributions effectively. Mathematically, Batch Normalization can be expressed as:

$$BatchNorm(x) = \frac{x - mean(x)}{sqrt(var(x))} * \gamma + \beta$$

where x represents the input batch, $mean(x)$ and $var(x)$ represent the mean and variance of the input, and γ and β are learnable parameters.

3.3.5 1D Maximum Pooling layer

The one-dimensional maximum pooling layers are used for downsampling the input. In the context of one-dimensional convolutional layers, a maximum pooling layer takes the maximum value over a certain window of inputs, effectively reducing the size of the inputs. This operation summarizes the most salient features within each window, preserving the most important information while discarding less prominent details. In our model, the pooling layers possess a pool size of 5 which means that the maximum is calculated using the 5 values closest to an input x (This includes x as well). The stride of this layer is 2 which indicates that the next input after x will be x + 2. Figure 8 demonstrates the pooling process. We utilize the same pooling and stride value as Khan et al. (2023) as we want our results to be comparable and our architectures are similar.

3.3.6 Flatten Layer

Following the convolutional and pooling layers, the output is then flattened. The Flatten layer reshapes the multidimensional output into a one-dimensional vector. This flattening of the data is necessary as the following Dense layers, which are fully connected layers, expect data in this format.

3.3.7 Dense Layer

Dense layers follow the Flatten layer, where each neuron in a Dense layer is connected to every neuron in the previous layer. These layers perform classification on the features extracted by the Conv1D layers and combined by the MaxPooling1D and Flatten layers. They help the network learn non-linear combinations of high-level features as they are fed into the final layer. The first two dense layers in the model make use of the ReLU activation function whereas the last layers uses the softmax function.

3.3.8 Softmax Output Layer

The last layer in our model is the Softmax layer, which is a type of activation function particularly useful for multi-class classification problems. The Softmax function outputs a vector that represents the probabilities of each class, with all probabilities summing up to 1. Mathematically, the Softmax function can be defined as:

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_{j=1}^{C} exp(x_i)}$$

where x_i represents the input to the softmax function for class i, and C represents the total number of classes. j is a variable that is used to get the probabilities of all classes in the set. The Softmax layer ensures that the output probabilities are normalized and can be interpreted as the likelihood of the input belonging to each class. This allows our model to effectively provide probabilities across the various ECG classes we are predicting, facilitating decision-making and result analysis.

3.4 Model Hyper-parameters

The configuration of a model that is initialized upon creation is known as a hyper-parameter. These values remain independent of the training data and do not change once initialized. This section illustrates the process used to determine the hyper-parameters of our model

3.4.1 Optimizer

Upon setting up the learning architecture, the model was compiled to initiate the training process. The choice of the optimizer is an important aspect of this phase, as it governs how the model's weights will be updated during the learning process. We chose the Adam optimizer for this task, an algorithm that stands out for its adaptive learning rate capabilities. Adam stands for Adaptive Moment Estimation, and its primary advantage is that it adjusts the learning rate dynamically for each weight in the model, depending on the computed gradients. This characteristic allows for faster convergence during training and enhances the generalization ability of the model, thereby improving the overall learning efficiency. Mathematically, the Adam optimizer updates the weights using a combination of adaptive learning rates and momentum terms. The learning rate adaptation is performed based on the moments of the gradients. A more comprehensive description of the Adam optimizer can be obtained from the creators of the algorithm (Kingma & Ba, 2014).

3.4.2 Loss Function

In multi-class classification problems like ours, the choice of loss function plays a crucial role in guiding the model to optimize its weights for accurate predictions. We selected the categorical cross-entropy loss function for this purpose, which is widely utilized in such scenarios. This function measures the dissimilarity between the predicted probability distribution and the true label distribution. It penalizes the model more when it assigns lower probabilities to the correct classes, thereby nudging it to predict higher probabilities for the correct class labels.

Mathematically, the categorical cross-entropy loss function can be defined as follows:

$$L = \sum_{i=1}^{C} y_i log(\hat{y_i})$$

Where:

1. C is the number of classes
2. y_i is the true label distribution for a class i
3. $\hat{y_i}$ is the predicted probability distribution for a class i
4. L is the loss value

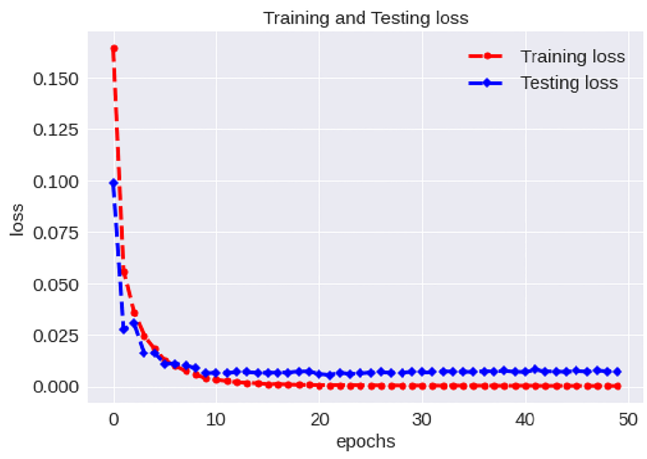

Figure 9: Loss over time of Khan et al. (2023)'s model. Reprinted from their paper. The loss plateaus by the 10th epoch.

By minimizing the categorical cross-entropy loss, the model aims to improve the alignment between its predicted probabilities and the true label distributions. This optimization objective guides the model to make more accurate predictions and learn the relationships between features and class labels effectively.

3.4.3 Epochs

An epoch is a complete pass through the training data. This means that the model is presented with all of the data in the training set, and the weights are updated accordingly. The number of epochs that a model is trained for can vary depending on the complexity of the model and the dataset. For our model, we determined the number of epochs needed by observing the loss trend observed in the model of Khan et al. (2023) since our architectures are similar. An examination of Figure 9 reveals that the loss plateaus around 10 epochs. We will this value to conserve time since the training of the model ended up being more intensive than we hoped for.

3.4.4 Batch Size

When training a model, instead of feeding in all the training data at once and updating the weights after one pass, training is done in batches. Thus, a batch of examples is fed to the model which calculates the loss and updates its weights accordingly. This technique massively speeds up training time since batches can be further split into smaller components that can be processed in parallel by a GPU. For our model, we selected a batch size of 512 which struck a balance between training time and accuracy. We made this decision when using Khan et al. (2023)'s observation that there was a small difference in accuracy between a batch size of 32 (98.72%) compared to a batch size of 500 (98.5%). We used Google Colab's free tier of GPU which disconnected after a few hours of inactivity so we selected the batch size of 500 to get around this limitation.

3.4.5 Learning Rate

Learning rate refers to how much to adjust the model's weights each time the model processes a batch of training data. It controls how rapidly the model learns from the data during training. A high learning rate means the model's weights are adjusted by a larger amount with each training step, resulting in faster learning but possibly overshooting the optimal weights. A low learning rate means smaller weight adjustments and slower learning but more stability. Khan et al. (2023) determined that a learning rate of 0.001 produced the best accuracy with their model. Therefore, we will use the same value as well.

3.4.6 Regularization

To prevent overfitting, we made use of regularization, specifically the L2-norm regularizer. It is calculated as the squared sum of all the model parameters, which gets added (after possible scaling) to the loss function. It penalizes models that have parameters with high absolute values. Preferring smaller absolute parameters forces the model to find a less complex solution which does not rely too heavily on the training data and represents a more general solution.

However, if the strength of regularization is too high, then we run into the risk of underfitting where the model is unable to learn useful features from the data.

In order to determine the optimal regularization strength for our model, we made use of stratified 5-fold cross-validation which preserved the distribution of classes in each fold. In this process, we further split the training data into 5 disjoint sets and trained the model on 4 sets while validating it on the remaining set. This process was repeated until all of the 5 sets were used as a validation set. We then averaged the validation scores across the 5 folds to get a less biased estimate of the model's performance.

This entire process was repeated for the 4 candidates of λ that we chose for the regularization strength (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}). The hyper-parameter values used during cross-validation were the same as described above. We tried other values of λ that differed by magnitudes, but these values led to extremely low validation accuracies which represent underfitting or overfitting. Figure 10 illustrates our findings with the 4 candidates. Our model demonstrated the highest validation accuracy with a regularization strength of 10^{-3}. Thus, we will set the regularization hyper-parameter to this value when training our model for the testing set. It should be noted that the difference between validation accuracy did not vary substantially between the lower λ values. Thus, we are not confident that the value of 10^{-3} is the best possible hyper-parameter for our model. We did not redo our cross-validation due to its intensive nature and to prevent p hacking.

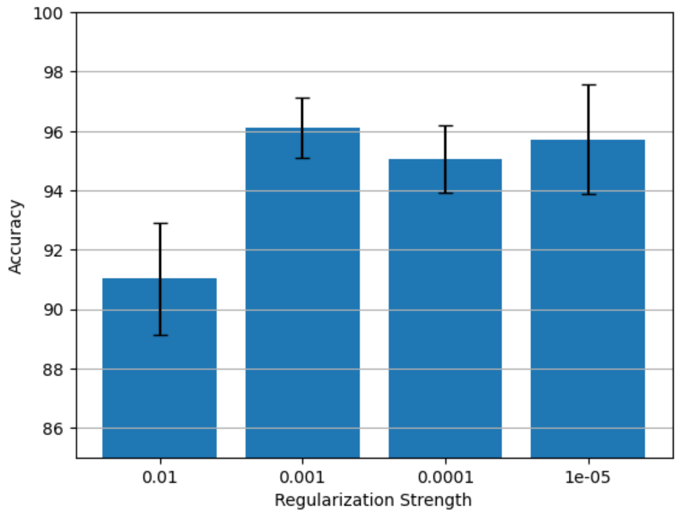

Figure 10: Plotting the accuracy of the model obtained through cross-validation for regularization values: 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}. The standard error is illustrated as well.

3.5 Model Evaluation

Now that the model has been created, we need to use some metrics to evaluate its performance. Accuracy (Acc) is a general-purpose metric which depicts how many predictions of the model are correct. However, in the field of medicine, more complicated metrics are used to determine the performance of machine learning models.

Sensitivity (Sen), also called recall or true positive rate, measures the proportion of actual positives that are correctly identified as such. It demonstrates the ability of a test to correctly identify those with the condition. A high sensitivity means that there are few false negatives, i.e. few patients with the disease are missed.

Specificity (Sp) measures the proportion of actual negatives that are correctly identified. It shows the ability of a test to correctly identify those without the condition. High specificity means that there are few false positives, i.e. few patients without the disease are incorrectly identified as having the disease.

Precision (Prec) tells us the ratio of true positives to all positive predictions. High precision means that there are few false positives, i.e. few incorrectly identified patients.

Listed below are the mathematical definitions of the above terms:

$$Sensitivity = \frac{TP}{TP + FN}$$
$$Specificity = \frac{TN}{TN + FP}$$
$$Precision = \frac{TP}{TP + FP}$$

Where TP = True Positives, FP = False Positives, TN = True Negatives, FN = False Negatives

4 Results

We tested our model on the part of the original dataset that has not been seen by the model during training or during the tuning of the hyperparameters. The metrics mentioned in the previous section are summarized in Table 2. We will also compare our metrics to those obtained by other papers which we cited in order to judge our model's performance.

Table 2: Results of our model compared with other ECG studies cited in our paper

Authors	**Model**	**Classes**	**Acc (%)**	**Prec (%)**	**Sen (%)**	**Sp (%)**
Mousavi & Afghah (2018)	Seq2Seq	4	99.92	99.26	98.66	99.7
Rahman Khan et al. (2020)	Deep 1D CNN	5	95.2	95.2	95.4	95
Khan et al. (2023)	Deep ResNet	5	98.63	92.86	92.41	99.06
Ours	Deep ResNet	5	96.09	96.21	96.01	99.05

Upon examination of our results, we observe that there is a fair difference between our model and Khan et al. (2023) which served as our reference. Our model possesses higher precision, sensitivity and specificity but lower accuracy. However, we can still attest to the power of the Deep ResNet architecture as our model outperforms the CNN by Rahman Khan et al. (2020). The state of the art is still far out of our reach as seen with Mousavi & Afghah (2018)'s model. Table 2 is not a comprehensive comparison against all techniques proposed in the field, nonetheless, it still offers insight into the efficacy of our model.

We were also interested in how the model performs in specific categories. This is important as the original dataset was highly imbalanced. Table 3 displays the evaluation of the model's performance on different classes. We observe that the precision and sensitivity of the F and S class are remarkably lower compared to the others. These were the minority classes which only accounted for around 3% of the dataset and indicate that our SMOTE data balancing was not perfectly effective at addressing the class imbalance problem.

Table 3: Metrics across different classes

Class	**Prec (%)**	**Sen (%)**	**Sp (%)**
F	70.48	75.46	99.76
N	99.76	96.91	94.59
Q	96.42	99.71	99.71
S	52.79	87.18	97.96
V	92.8	88.69	99.48

To get a more detailed picture of the model's performance, we can take a look at the confusion matrix in Figure 11. This gives information about which labels are often misclassified as which other labels. The most frequent misclassifications occurred with the F and S minority classes which were falsely predicted as the majority N class.

5 Discussion

Our study focused on developing and evaluating a Deep ResNet model for the classification of ECG signals. We have taken Khan et al. (2023) as our main inspiration and modified their architecture. The model is able to identify normal

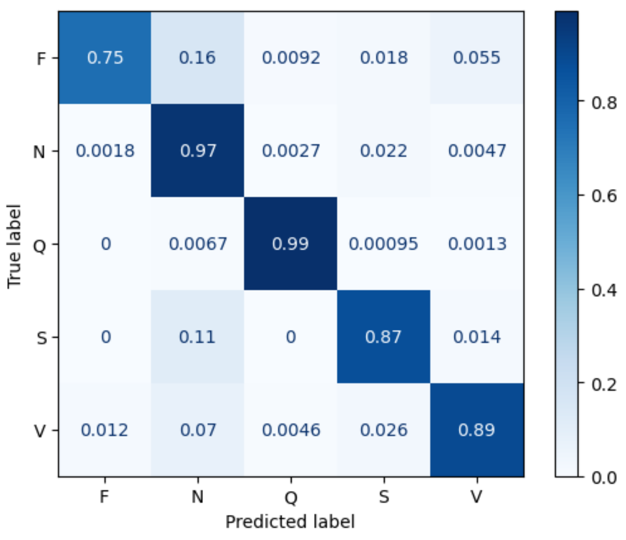

Figure 11: Confusion matrix on the model's performance on different labels

and pathological heartbeats from the ECG signals with an overall accuracy of over 96%. Utilizing the classification capabilities of a CNN and the provided large annotated dataset, the model has shown that a deep residual CNN works well for this purpose. This suggests that our model has the potential to assist healthcare professionals in the timely and accurate diagnosis of cardiovascular conditions.

However, it is important to acknowledge the limitations of our project. One such limitation is the reliance on the specific dataset. The model may produce lower accuracy and precision rates for different populations and data. Over 80 percent of the MIT BIH dataset used to train and test our model is made up of normal heartbeats. This makes the model better at classifying normal heartbeats as opposed to pathological ones. In a medical diagnosis, this can lead to problems or misguidance of professionals.

Another limitation of the model is that even though we tried to overcome the unbalanced data problem by making use of the SMOTE algorithm, it only correctly classified 75% of the cases labelled 'F'. It also struggled with labels 'S', and 'V', they were classified correctly only in 87 and 89 percent of the cases respectively. This could be because 'F' is heavily underrepresented in the training dataset and the other two categories also make up only a fraction of it (as shown in Table 1). For all of the three mentioned categories, when the model made a mistake, it classified them as normal most of the time which could be because the training dataset was mostly containing normal heartbeats. While the SMOTE algorithm can help in dealing with unbalanced datasets in our case, it showed that the model learned to recognize the categories the best when they were already well-represented in the training data. In medical diagnosis, it is crucial to keep an eye on false negatives as failing to recognise a condition could have fatal consequences. If we were to test our model on a dataset containing an equal number of cases from each category, we would have attained an abysmal score since there would be no bias towards the majority class.

There is also a substantial difference in the performance of our model compared to Khan et al. (2023) even though our architectures are similar. We have reason to believe that this difference stems from our pre-processing of data since we relied on other sources for implementation. The pre-processing section in the paper is quite brief and we were unable to properly understand all of the steps used by the authors. We did not find evidence that our modification to the architecture reduced our accuracy since our model was unable to reach the reported accuracy of 98.63% even without the regularization that we added.

In the same vein, it is important to note that our testing of hyper-parameters was not rigorous. We utilized values from Khan et al. (2023) to conserve time and only experimentally determined the regularization strength. Even with this shortcut, the cross-validation process required around 2 hours to complete which served as a challenge since Google Colab would disconnect the runtime after an hour of inactivity. In hindsight, it would have been better to use a simpler architecture that would take less time to train and permit us greater flexibility. We are particularly curious about the usage of different types of regularization norms and the benefit of finetuning the model on individual data.

To conclude, our model demonstrates the effectiveness of deep residual CNN models for classification. This model could be improved by making the model bigger. Most importantly the model could be trained and tested with more data and from more diverse datasets to improve its performance.

Moving forward, it is important to integrate similar networks into the healthcare system. It would not only provide assistance for the healthcare professionals but it would also help make better models. The healthcare system could provide more data and computing power to make improved models. Professionals could also provide feedback on the reliability and feasibility of such models. Investigating the interpretability and explainability of machine-learning methods and models could also help enhance their clinical acceptance.

REFERENCES

Association for the Advancement of Medical Instrumentation (AAMI). *Testing and Reporting Performance Results of Cardiac Rhythm and ST-segment Measurement Algorithms*. ANSI/AAMI/ISO. American National Standards Institute, Inc. (ANSI), 1998-2008. ISBN 9781570201165.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, jun 2002. ISSN 1076-9757.

Ary L. Goldberger, Zachary D. Goldberger, and Alexei Shvilkin. Chapter 3 - how to make basic ecg measurements. In Ary L. Goldberger, Zachary D. Goldberger, and Alexei Shvilkin (eds.), *Goldberger's Clinical Electrocardiography (Ninth Edition)*, pp. 11–20. Elsevier, ninth edition edition, 2018. ISBN 978-0-323-40169-2. doi: https://doi.org/10.1016/B978-0-323-40169-2.00003-2. URL https://www.sciencedirect.com/science/article/pii/B9780323401692000032.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.90. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90.

Mikael Häggström. Schematic representation of the qrs complex., 2010. URL https://en.wikipedia.org/wiki/QRS_complex#/media/File:QRS_complex.svg. [Online; accessed June 29, 2023].

Fahad Khan, Xiaojun Yu, Zhaohui Yuan, and Atiq ur Rehman. Ecg classification using 1-d convolutional deep residual neural network. *PLOS ONE*, 18(4):1–22, 04 2023. doi: 10.1371/journal.pone.0284791. URL https://doi.org/10.1371/journal.pone.0284791.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Nastaran Mohammadian Rad. *Deep Learning for Abnormal Movement Detection using Wearable Sensors: Case Studies on Stereotypical Motor Movements in Autism and Freezing of Gait in Parkinson's Disease*. PhD thesis, 05 2019.

G.B. Moody and R.G. Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001. doi: 10.1109/51.932724.

Sajad Mousavi and Fatemeh Afghah. Inter- and intra- patient ecg heartbeat classification for arrhythmia detection: A sequence to sequence deep learning approach. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1308–1312, 2018.

Jiapu Pan and Willis J. Tompkins. A real-time qrs detection algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3):230–236, 1985. doi: 10.1109/TBME.1985.325532.

Mohammad Mahmudur Rahman Khan, Md. Abu Bakr Siddique, Shadman Sakib, Anas Aziz, Abyaz Kader Tanzeem, and Ziad Hossain. Electrocardiogram heartbeat classification using convolutional neural networks for the detection of cardiac arrhythmia. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 915–920, 2020. doi: 10.1109/I-SMAC49090.2020.9243474.

Kushagra Sharma. Pan tompkins qrs detection, 2021. URL https://github.com/antimattercorrade/Pan_Tompkins_QRS_Detection.

World Health Organization [WHO], 2023. URL https://www.who.int/health-topics/cardiovascular-diseases.