Aditya Kendre
Cumberland Valley HS
12th Grade
Year 5

PhonoNet:

Generative Adversarial Networks for PCG Arrhythmia Detection

Objective

Introduction

An estimated three **million cases** of **arrhythmia** occur in the United States yearly (Mayo Clinic), with **300,000 sudden deaths** per year – an incidence rather higher than stroke, lung cancer, or breast cancer (American Heart Association). Traditionally, non-invasive arrhythmia analysis is based on multiple electrodes that reflect the electrical activity on ECGs. This method, despite being accurate, limits the use case to hospitals and clinics with specialized equipment; thus, limiting the portability of diagnosing, let alone classification of the type of pathology.

Problem

Current detection methods have **limited performance** in **pathologies and lack real-time** classification capabilities.

Motivation

To create a **fast** and **accurate** model capable of detecting Cardiovascular modalities, specifically arrhythmias in heart sound recordings (PCGs) without the need for specialized equipment.

Question

Is it possible to use **Generative Adversarial Networks** (GANs) to accurately detect arrhythmias in PCGs and surpass previous methods in detection tasks?

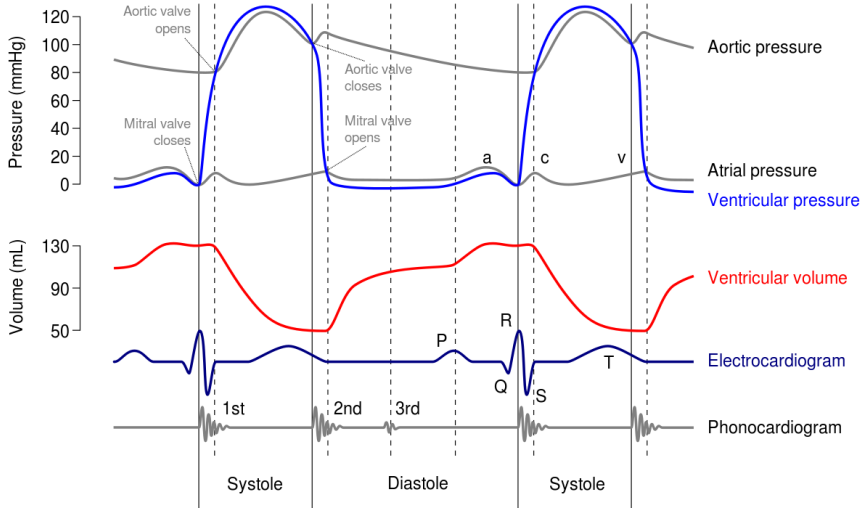
Hypothesis

If a Generative Adversarial Network is used to create spurious data, then the model will outperform previous state-of-the-art methods in classification, because the specious data will aid the model in extracting significant features from a ground truth dataset.

Background & Engineering Goals

Phonocardiograms (PCGs) are sounds that are created by the mechanical movement of the heart. This physical movement produces four distinct sounds: S1, S2, S3, S4, and murmurs. S1 and S2 are sounds created by a healthy heart; whereas S3, S4 and murmurs refer to diseases or anomalies.

In diagnosing heart sounds, two major challenges arise: localization and classification. **Localization** aims to find the position of the biomarkers in heart sounds. By doing this, heart sounds can be segmented into signals containing a single heart sound. Furthermore, **classification** attempts to categorize heart sounds into normal and abnormal groups by exploiting the information extracted from localization.



Conventional heart sound localization and classification methods involve time, frequency, or both, and are typically dependent on machine learning algorithms to enhance the results. These algorithms typically include **artificial neural networks** (ANNs), **support vector machines** (SVMs), **self-organizing maps** (SOMs), and are limited to the number of samples and pathologies covered in a given dataset. Often, large datasets that accurately represent real-world recordings only contain two categories: normal and abnormal. This leads to a surface-level analysis of the heart sounds.

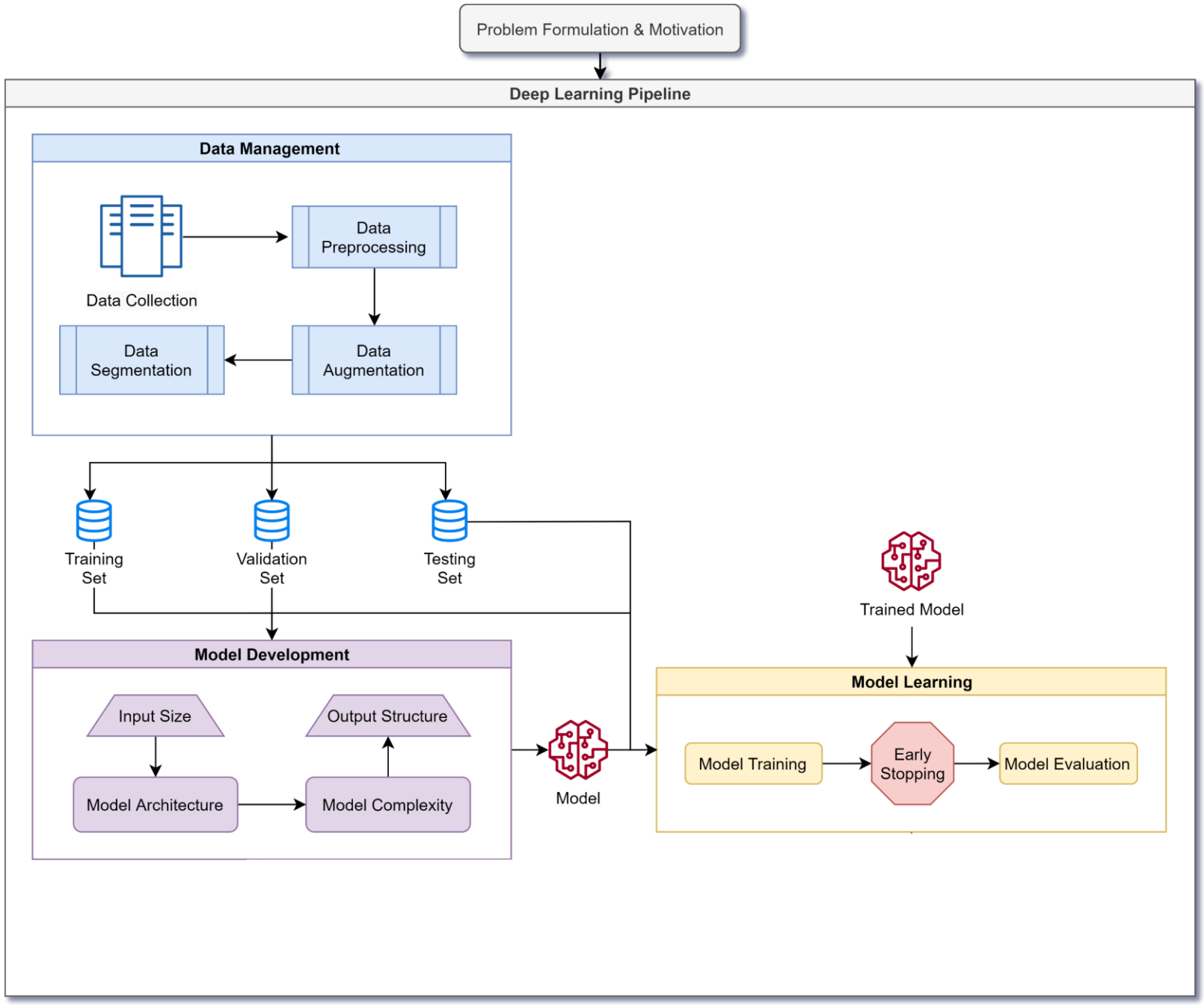
From the viewpoint of practical applications, the development of computationally efficient solutions is extremely important to the success of a model's deployment. Many studies have negated to comment on the practicality of their proposed methods. From our research, we have concluded only two studies have noted their time efficiency, Fernando et al. and Messner et al. The fastest model processed 1000 heart state localizations in 56.88 seconds (Fernando et al.), suggesting the model can process **18 bps or 1054 bpm**. Although the proposed model is able to achieve near real-time performance, these results are excluding the classification of heart arrhythmias, and only include the localization of S1, S2 markers.

Thus, the problem of computationally efficient and accurate classification of noisy heartbeats remains a problem.

Solution

The new semi-supervised approach is composed of **two subsystems**; the first subsystem uses a **discriminator**, in which PCG signals are classified into categories based on arrhythmias in the signal. The second subsystem – a **generator**, aims to generate data such that, when fed into the discriminator, the discriminator will classify the generated data as abnormal or normal. While the discriminator aims to simultaneously classify the generated data as fake and classify the real PCG signals to their respective categories.

Methods


Problem Formulation & Motivation
Deep Learning Pipeline
Data Management
Data Collection
Data Preprocessing
Data Augmentation
Data Segmentation
Training Set
Validation Set
Testing Set
Model Development
Input Size
Output Structure
Model Architecture
Model Complexity
Model
Trained Model
Model Learning
Model Training
Early Stopping
Model Evaluation

Data Management

Data Collection

Although PCG signals are analyzed less often than ECG signals, these signals are rather analyzed in real-time by physicians and healthcare workers. Preliminary studies done on PCG segmentation and classification primarily used private datasets. Hence, there existed no publicly available datasets until recently. Since then, many public datasets have been developed aiding researchers in their studies and creating open benchmarks for researchers to use in comparing similar findings. However, these datasets are still limited by the number of classes that are collected, when compared to ECG datasets.

Currently, only three major supervised PCG datasets exist: **PhysioNet Classification of Heart Sound Recording Challenge dataset**, **PASCAL Heart Sound Challenge dataset**, and the Heart Sound and Murmur Library. These datasets are all anonymized and de-identified for the safety of their subjects, and thus includes no personal information such as name, income, age, etc.

Data augmentation

Data augmentation is a strategy that enables a significant increase in the diversity of data available while training a model, without actually collecting new data. Data augmentation techniques aim to slightly alter existing data to a point where the model cannot recognize the augmented data as one it has trained on before, but still retains the characteristics of the data's category. This helps in reinforcing important features within the data and is only done during the training portion of the workflow.

We resample the heart sound recordings to different frequencies to simulate slower and faster beats per minute (bpm). The normal bpm for a human is between 60-100 bpm. Thus, measuring the sample distance between the first S1 (the start of systole) and the second marker, S2, we calculate the bps and resample accordingly.

Data Segmentation

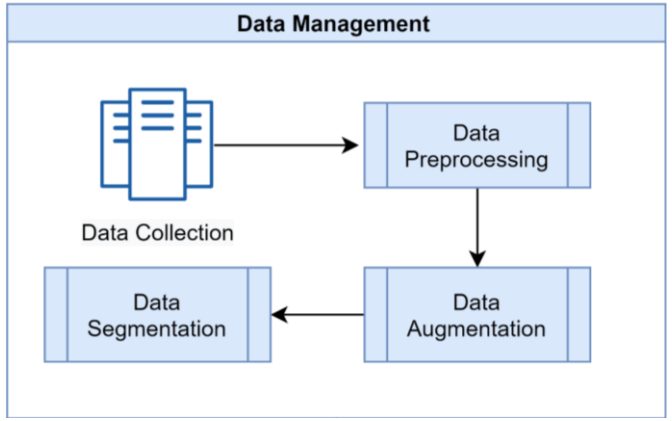
Data segmentation refers to the process of creating cross-validation datasets. This process assists in validating if the model is overfitting to the dataset. These datasets include the training set, validation set, and testing set. Typically, the training set is 70%-80% of the dataset, the reset of the dataset is split among the validation set and testing set. Here, we split the data 80% training, 10% validation, and 10% testing.

Data Preprocessing

PCG recordings often are recording in non-ideal environments that are filled with unwanted background noise and interference. Data preprocessing is the process of altering the data in the signal, often by denoising, normalizing, standardizing, and transforming the signal. These steps are crucial for automatic localization and classification tasks. Preprocessing the data allows a model to extract meaning features efficiently and reveals the physiological structure of the heart sounds. Furthermore, preprocessing helps ensure that the data that is fed into the model is always in the same domain. This allows the model to generalize more easily.

We first resample the data to **500 Hz**, to decrease the spatial resolution of the heart sound recordings for easier processing, but still dense enough to retain important features. Thus, helping the model to converge faster. The resampled data is **standardized using the standard score**:

$$z_n = \frac{x_n - \mu}{\sigma}$$

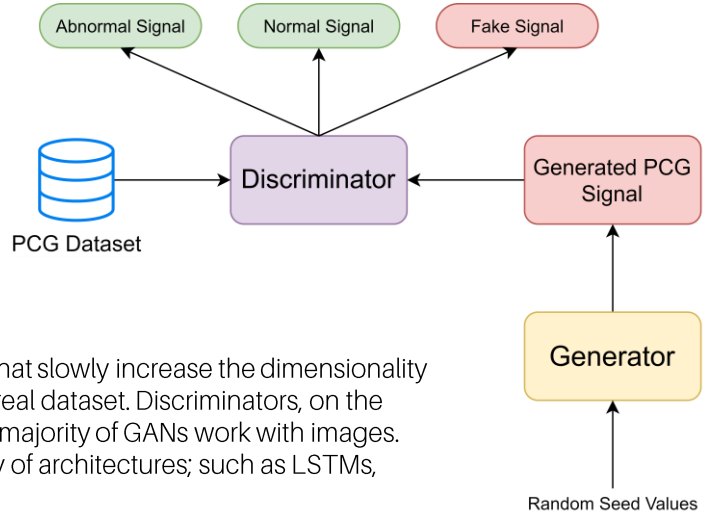


Model Development

Model Architecture

Here we propose using Generative Adversarial Networks (GANs) for increased success in PCG heart sound detection. GANs poses a unique advantage over traditional machine learning and deep learning methods, in that a model learns to mimic a dataset by creating its own data and tries to fool a discriminator into thinking the generated data is real.

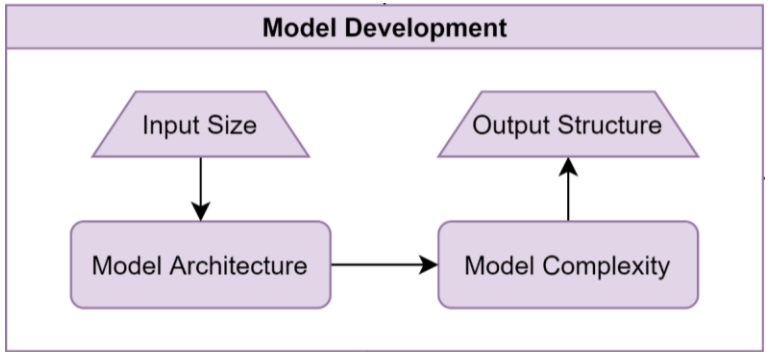
In a supervised approach, a GAN consists of two parts, a generator and a discriminator. The generator is responsible for creating fake heart sound data, while the discriminator tries to predict whether the incoming data is fake or real. In a semi-supervised approach, however, the discriminator is fed data from a real dataset with multiple classes and the generated data from the generator. Here, the discriminator tries to classify the generator's fake data, as well as predict the classes from the real dataset.



Model Complexity

Traditionally, generators are dense layers that slowly increase the dimensionality of the generated data to match that of the real dataset. Discriminators, on the other hand, are commonly CNNs because majority of GANs work with images. However, it is possible to use a wide variety of architectures; such as LSTMs, RNNs, SVMs, DNNs, ANNs, Transformers.

As mentioned above, there are many types of model architecture, some are used for classification, and others for feature extraction. Optimizing the combination of feature extraction layers and classification layers is extremely time consuming and computationally taxing. This is because there exist many combinations of hyperparameters, thus making it difficult to optimize each parameter. To optimize hyperparameters, we used hyperparameter sweeps to make the optimization process more efficient. This method involves using one of three methods: grid search, random search, and Bayesian search. Grid search computes each possible combination of all hyperparameters and tests them all. Although this is very effective, it can be computationally costly. Random search selects a new combination at random, provided a distribution of values. This method is surprisingly effective and scales very well. Bayesian search creates a probabilistic model of metrics and suggests parameters that have a high probability of improving metrics. This works well for small scale projects, but scales poorly as the complexity of parameter relationships increase. Here, we used random search to optimize our hyperparameters.



Model Learning

Model Training

During the training phase, the model is trained using backpropagation in conjunction with a cost function. Backpropagation attempts to calculate the gradient of the cost function with respect to the weight and biases of the model. This process involves an optimizer, which optimizes the model's parameters and a cost function that measure the correctness or incorrectness of the model. The goal of the optimizer is to minimize the cost function's error by adjusting the parameters to the given label. In this study, we used the Adam optimizer in union with Cross-Entropy Loss. The Adam optimizer uses a hyperparameter that dictates the change in the model's parameters on each backpropagation step, this is called the learning rate. Here we choose a learning rate of 0.0001.

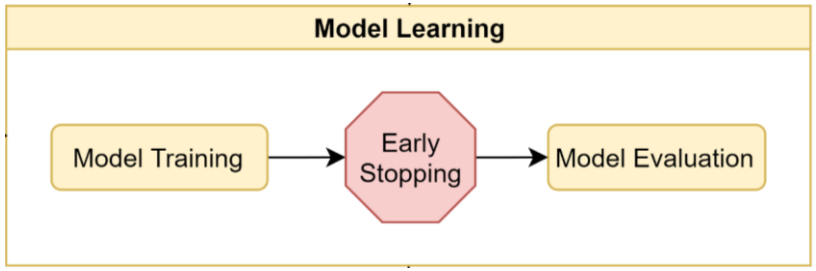
The model is only trained on the training set; thus, backpropagation only occurs on the training set. Additionally, for each step in the training set, the optimizer backpropagates and optimizes the parameters and calculate metrics to further evaluate the model. The number of steps in the training set is dictated by the batch size, the number of signals the model is trained on, in a single forward pass. Here we use a batch size of 32, meaning that the model is fed 32 signals per input. This significantly speeds up the process of training as more signals are passed through the model every time the model is optimized. A full pass of the training set is called an Epoch, here we train the model on 100 Epochs.

Early Stopping

To ensure the model is not overfitting, but generalizing to the training set, we use a validation set to track the metrics of the model. In theory, the metrics on the training set equal to that of the validation set. In practicality,
after many epochs of training the metrics of the validation set become static, but the metrics of the training set still increase. This suggests that the model is overfitting. Thus, we stop training the model on the training set and test it on the testing set.

Model Evaluation

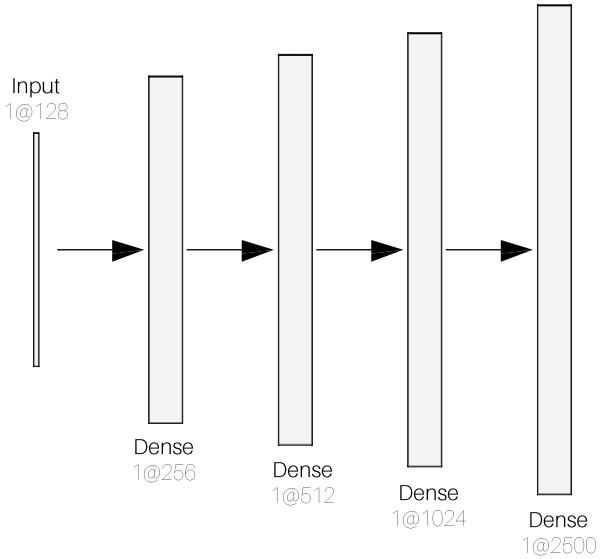
Testing sets or hold-out sets are used to validate the metrics of a model, this is because both the validation set and the testing set have been tested by the model; thus, the model has developed a latent bias to both sets. Therefore, a third set is needed to assess the model's ability to generalize on an independent dataset.



Architecture

Generator

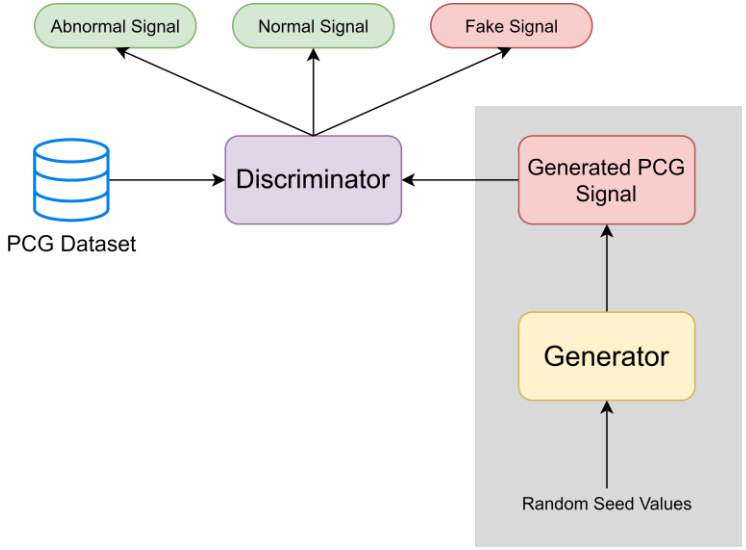
The **input** consists of a 2-dimensional tensor (batch size x signal) with a length of 128. This input is randomly generated.



The **Dense** layers upscale the length of the input vector until the length reaches that of a real input signal (2500). This allows for the output of the generator to be directly fed into the discriminator.

The Rectified Linear activation function alters the range of the incoming data by setting all numbers below 0 to 0 and leaving all positive numbers intact.

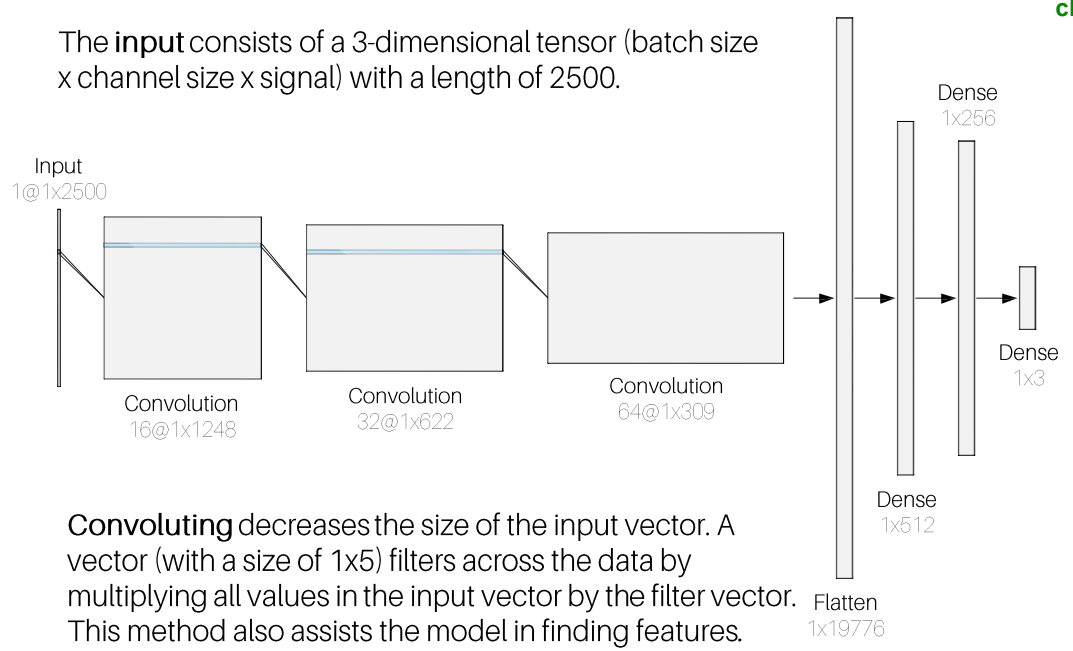
```
class Generator(pblm.PrebuiltLightningModule):
    def __init__(self, denoising=False):
        super().__init__(self.__class__.__name__)

        def block(in_feat, out_feat, normalize=True):
            layers = [nn.Linear(in_feat, out_feat)]
            if normalize:
                layers.append(nn.BatchNorm1d(out_feat, 0.8))
                layers.append(nn.LeakyReLU(0.2, inplace=True))
            return layers

            self.model = nn.Sequential(
                        *block(128, 256, normalize=False),
                        *block(256, 512),
                        *block(512, 1024),
                        nn.Linear(1024, 2500),
                        nn.Tanh()
                        )
    def forward(self, z):
        z = self.model(z)
    return z
```



Architecture

Discriminator

The **input** consists of a 3-dimensional tensor (batch size x channel size x signal) with a length of 2500.



Convoluting decreases the size of the input vector. A vector (with a size of 1x5) filters across the data by multiplying all values in the input vector by the filter vector. This method also assists the model in finding features.

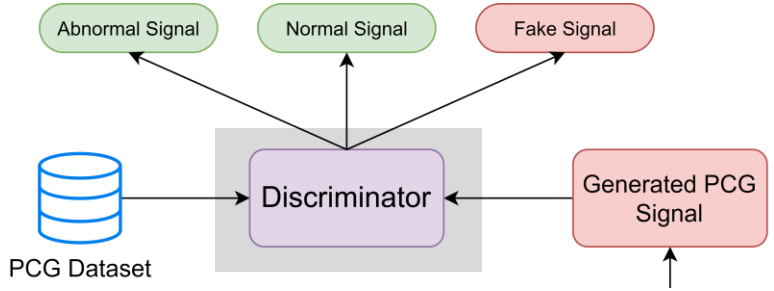
The **Rectified Linear activation** function alters the range of the incoming data by setting all numbers below 0 to 0 and leaving all positive numbers intact.

The **liner function** flattens the incoming result (batch size x 64 x 309) into a 2D tensor (batch size x19776).

The **Dense** layers downscales the length of the input vector until the length reaches that of the number of classes. This allows for the output of the discriminator to be directly interpreted by the cost function.

The **Linear Output** layer transforms the Linear layer output into a 1x3. Each column in the tensor represents a class's likelihood of being the correct class in the dataset. Thus, the column with the largest values is the model prediction for the input

```
class Discriminator(pblm.PrebuiltLightningModule):
  def __init__(self, denoising=False):
    super().__init__(self.__class__.__name__)

    # Model Layer Declaration
    self.conv1 = nn.Conv1d(1, 16, kernel_size=5, stride=2)
    self.conv2 = nn.Conv1d(16, 32, kernel_size=5, stride=2)
    self.conv3 = nn.Conv1d(32, 64, kernel_size=5, stride=2)
    self.dense1 = nn.Linear(64*309, 512)
    self.dense2 = nn.Linear(512, 256)
    self.dense3 = nn.Linear(256, 3)

  def forward(self, x):

    x = x.reshape(x.shape[0], 1, -1)

    # Convolutional Layer
    x = self.conv1(x)
    x = nn.functional.relu(x)
    x = self.conv2(x)
    x = nn.functional.relu(x)
    x = self.conv3(x)
    x = nn.functional.relu(x)

    # Flattening
    x = x.reshape(x.shape[0], -1)

    # Dense Layers
    x = self.dense1(x)
    x = nn.functional.relu(x)
    x = self.dense2(x)
    x = nn.functional.relu(x)
    x = self.dense3(x)

    return x
```



Raw Results

	Specificity	Sensitivity	Accuracy
Trials 1	0.9375	0.971429	0.955224
Trials 2	0.955224	1	0.977612
Trials 3	0.898551	1	0.947761
Trials 4	0.935484	1	0.970149
Trials 5	0.967213	1	0.985075
Trials 6	0.90625	1	0.955224
Trials 7	0.955224	1	0.977612
Trials 8	0.855072	1	0.925373
Trials 9	0.935484	1	0.970149
Trials 10	0.967213	1	0.985075
Trials 11	0.90625	1	0.955224
Trials 12	0.955224	1	0.977612
Trials 13	0.811594	1	0.902985
Trials 14	0.935484	1	0.970149
Trials 15	0.967213	1	0.985075
Trials 16	0.90625	1	0.955224
Trials 17	0.955224	1	0.977612
Trials 18	0.884058	1	0.940299
Trials 19	0.935484	1	0.970149
Trials 20	0.967213	1	0.985075
Trials 21	0.921875	1	0.962687
Trials 22	0.955224	1	0.977612
Trials 23	0.855072	1	0.925373
Trials 24	0.935484	1	0.970149
Trials 25	0.967213	1	0.985075
Trials 26	0.90625	1	0.955224
Trials 27	0.955224	1	0.977612
Trials 28	0.768116	1	0.880597
Trials 29	0.935484	1	0.970149
Trials 30	0.967213	1	0.985075
Trials 31	0.90625	1	0.955224
Trials 32	0.790323	1	0.902985
Trials 33	0.806452	1	0.910448
Trials 34	0.847458	1	0.932836
Trials 35	0.955224	1	0.977612
Trials 36	0.847458	1	0.932836
Trials 37	0.917808	1	0.955224
Trials 38	0.885246	1	0.947761
Trials 39	0.915493	1	0.955224
Trials 40	0.768116	1	0.880597
Trials 41	0.915493	1	0.955224
Trials 42	0.944444	1	0.970149
Trials 43	0.90625	1	0.955224
Trials 44	0.891892	1	0.940299
Trials 45	0.851351	1	0.91791
Trials 46	0.955224	1	0.977612
Trials 47	0.955224	1	0.977612
Trials 48	0.876923	0.869565	0.873134
Trials 49	0.876923	0.956522	0.91791
Trials 50	0.885246	1	0.947761
Trials 51	0.855072	1	0.925373
Trials 52	0.815385	1	0.910448
Trials 53	0.90625	1	0.955224
Trials 54	0.935484	0.902778	0.91791
Trials 55	0.896552	1	0.955224
Trials 56	0.896552	1	0.955224
Trials 57	0.967213	1	0.985075
Trials 58	0.90411	1	0.947761
Trials 59	0.90411	1	0.947761
Trials 60	0.90625	1	0.955224

	Specificity	Sensitivity	Accuracy
Trials 61	0.905405	1	0.947761
Trials 62	0.945946	1	0.970149
Trials 63	0.955224	1	0.977612
Trials 64	0.876923	0.956522	0.91791
Trials 65	0.876923	0.956522	0.91791
Trials 66	0.927536	1	0.962687
Trials 67	0.815385	1	0.910448
Trials 68	0.815385	1	0.910448
Trials 69	0.935484	1	0.970149
Trials 70	0.896552	1	0.955224
Trials 71	0.955224	1	0.977612
Trials 72	0.896552	1	0.955224
Trials 73	0.967213	1	0.985075
Trials 74	0.90411	1	0.947761
Trials 75	0.768116	1	0.880597
Trials 76	0.90411	1	0.947761
Trials 77	0.921875	1	0.962687
Trials 78	0.905405	1	0.947761
Trials 79	0.935484	1	0.970149
Trials 80	0.905405	1	0.947761
Trials 81	0.723077	1	0.865672
Trials 82	0.9375	0.971429	0.955224
Trials 83	0.955224	1	0.977612
Trials 84	0.898551	1	0.947761
Trials 85	0.935484	1	0.970149
Trials 86	0.967213	1	0.985075
Trials 87	0.90625	1	0.955224
Trials 88	0.955224	1	0.977612
Trials 89	0.855072	1	0.925373
Trials 90	0.935484	1	0.970149
Trials 91	0.967213	1	0.985075
Trials 92	0.90625	1	0.955224
Trials 93	0.955224	1	0.977612
Trials 94	0.811594	1	0.902985
Trials 95	0.935484	1	0.970149
Trials 96	0.967213	1	0.985075
Trials 97	0.90625	1	0.955224
Trials 98	0.955224	1	0.977612
Trials 99	0.884058	1	0.940299
Trials 100	0.935484	1	0.970149
Trials 101	0.967213	1	0.985075
Trials 102	0.921875	1	0.962687
Trials 103	0.955224	1	0.977612
Trials 104	0.855072	1	0.925373
Trials 105	0.935484	1	0.970149
Trials 106	0.967213	1	0.985075
Trials 107	0.90625	1	0.955224
Trials 108	0.955224	1	0.977612
Trials 109	0.768116	1	0.880597
Trials 110	0.935484	1	0.970149
Trials 111	0.967213	1	0.985075
Trials 112	0.90625	1	0.955224
Trials 113	0.790323	1	0.902985
Trials 114	0.806452	1	0.910448
Trials 115	0.847458	1	0.932836
Trials 116	0.955224	1	0.977612
Trials 117	0.847458	1	0.932836
Trials 118	0.917808	1	0.955224
Trials 119	0.885246	1	0.947761
Trials 120	0.915493	1	0.955224

	Specificity	Sensitivity	Accuracy
Trials 121	0.768116	1	0.880597
Trials 122	0.915493	1	0.955224
Trials 123	0.944444	1	0.970149
Trials 124	0.90625	1	0.955224
Trials 125	0.891892	1	0.940299
Trials 126	0.851351	1	0.91791
Trials 127	0.955224	1	0.977612
Trials 128	0.955224	1	0.977612
Trials 129	0.876923	0.869565	0.873134
Trials 130	0.876923	0.956522	0.91791
Trials 131	0.885246	1	0.947761
Trials 132	0.855072	1	0.925373
Trials 133	0.815385	1	0.910448
Trials 134	0.90625	1	0.955224
Trials 135	0.935484	0.902778	0.91791
Trials 136	0.896552	1	0.955224
Trials 137	0.896552	1	0.955224
Trials 138	0.967213	1	0.985075
Trials 139	0.90411	1	0.947761
Trials 140	0.90411	1	0.947761
Trials 141	0.90625	1	0.955224
Trials 142	0.905405	1	0.947761
Trials 143	0.945946	1	0.970149
Trials 144	0.955224	1	0.977612
Trials 145	0.876923	0.956522	0.91791
Trials 146	0.876923	0.956522	0.91791
Trials 147	0.927536	1	0.962687
Trials 148	0.815385	1	0.910448
Trials 149	0.815385	1	0.910448
Trials 150	0.935484	1	0.970149

The table shows the **accuracy, sensitivity, and specificity** of 150 trials of training the GAN model on the dataset. The metrics displayed are the results of the testing set.

Accuracy: percent of correctly identified normal and abnormal heart sounds.

Sensitivity: percent of correctly identified abnormal heart sound recording from a sample of abnormal heart signals.

Specificity: percent of correctly identified normal heart sounds recordings from a sample of normal heart signals.

Note: The higher the value, the better

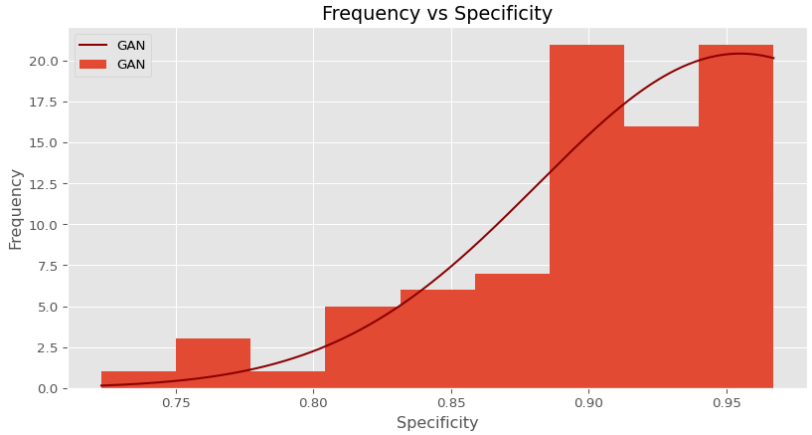
Analysis

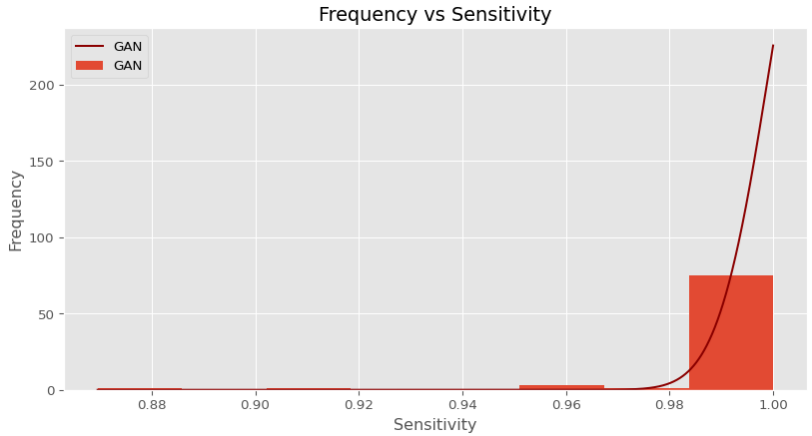
The graphs on the right illustrate that the GAN's accuracy, specificity, and sensitivity are skewed to the left. This suggests that the model is extremely successful at differentiating between abnormal and normal heart sounds.

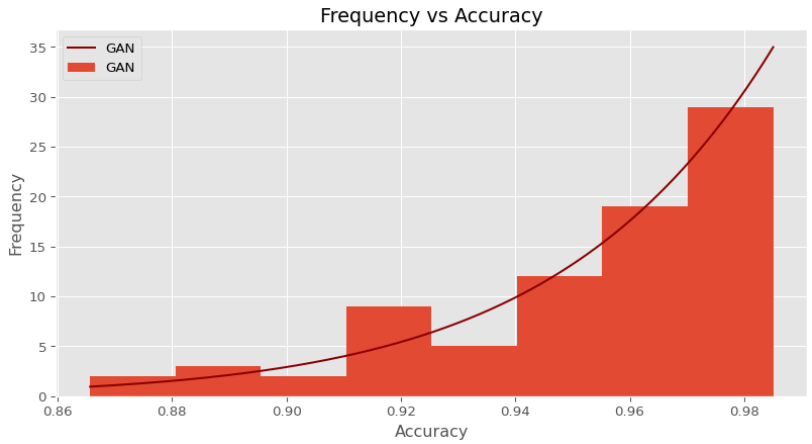
The specificity distribution has a mean of **0.9030**, with a standard deviation of **0.0547**. The best model reached a specificity of **0.9672**. This suggests that ~**90.3%** of normal heart sounds were correctly identified.

The sensitivity distribution has a mean of **0.9952**, with a standard deviation of **0.0197**. The best model reached a sensitivity of **1.0**. This suggests that **99.5%** of abnormal heart sounds were correctly identified. In detecting pathologies in medicine, we often attempt to maximize sensitivity, the rate at which a subject with a disease is correctly identified amongst other ill subjects. This is because we want to ensure that all potential subjects with a disease are sent for further examination. Essentially, weighting sensitivity higher than specificity, the rate of correctly identified normal or healthy patients from a sample of healthy patients. Thus, from the results, the proposed model is robust, in that it can detect abnormalities nearly ~100% of the time.

The accuracy distribution has a mean of **0.9498**, with a standard deviation of **0.0293**. The best model reached an accuracy of **0.9875**. This suggests that ~**95.0%** of both abnormal and normal sounds were classified correctly.







Related Woks

Study	Classification techniques	Beat types	Dataset	Time Efficiency	Results
Nogueira et al.	Classification of heat images generated from MFCC and time features using SVM, KNN, CNN, and random forest	Normal and abnormal heart sound	PhysioNet CinC challenge, 2016	-	Sensitivity: 96.47% Specificity: 72.65% Overall Score: 84.56%
Potes et al.	Time, spectrum, MFCC- based features, CNN, AdaBoost-Abstain classifier, and ensemble classifier	Normal and abnormal heart sounds	PhysioNet CinC challenge, 2016	-	Sensitivity: 77.81% Specificity: 92.42% Overall Score: 86.02%
Ortiz et al.	SVM using time, MFCC, DWT, and wavelet	Normal and abnormal heart sounds	PhysioNet CinC challenge, 2016	-	Test Score: 82.4%
Tang et al.	ANN using time, kurtosis, MFCC, energy, power spectrum cyclostationary features	Normal and abnormal heart sounds	PhysioNet CinC challenge, 2016	-	Sensitivity: 80.7-81.2%, Specificity: 82.9- 85% Overall Score: 81.8- 83.6%
Rubin et al.	CNN using heat maps generated using MFCC features	Normal and abnormal heart sounds	PhysioNet CinC challenge, 2016	-	Specificity: 95% Sensitivity: 73% Overall Score: 84%
Singh et al.	Time domain, frequency domain, cepstrum, statistical features, Bayes Net, nave Bayes, GSD, and Logit Boost	Normal heart sounds and murmurs	PASCAL Datasets: A & B	-	Specificity: 93.3% Sensitivity: 93.3% Accuracy: 93.3%
Krishnan et al.	Embedding Layers, CNN, DNN	Normal and abnormal heart sounds	PhysioNet CinC challenge, 2016	-	Sensitivity: 86.73% Specificity: 84.75% Accuracy: 85.65% Precision: 82.52% F1 Score: 84.58%
Fernando et al.	BiLSTM	S1, S2, None	PhysioNet CinC challenge 2016	17 classifications per second	Sensitivity: 97.2% Specificity: 97.5% Accuracy: 96.9% PPv: 96.3% F1 Score: 96.7%
Messner et al.	RNN	S1, S2, None	PhysioNet CinC challenge, 2016	8 classifications per second	Sensitivity: 96.1% Specificity: 95.1% F1 Score: 95.6%
Proposed Method	**GAN with CNN**	**Normal and abnormal heart sounds**	**PhysioNet CinC challenge 2016 & PASCAL Datasets: A & B**	**~1000 classifications per second**	**Sensitivity: 100.0% Specificity: 96.67% Accuracy: 98.75%**

Statistical Significance

t-test

P(T<=t) one-tail:

Specificity:

1.65E-10

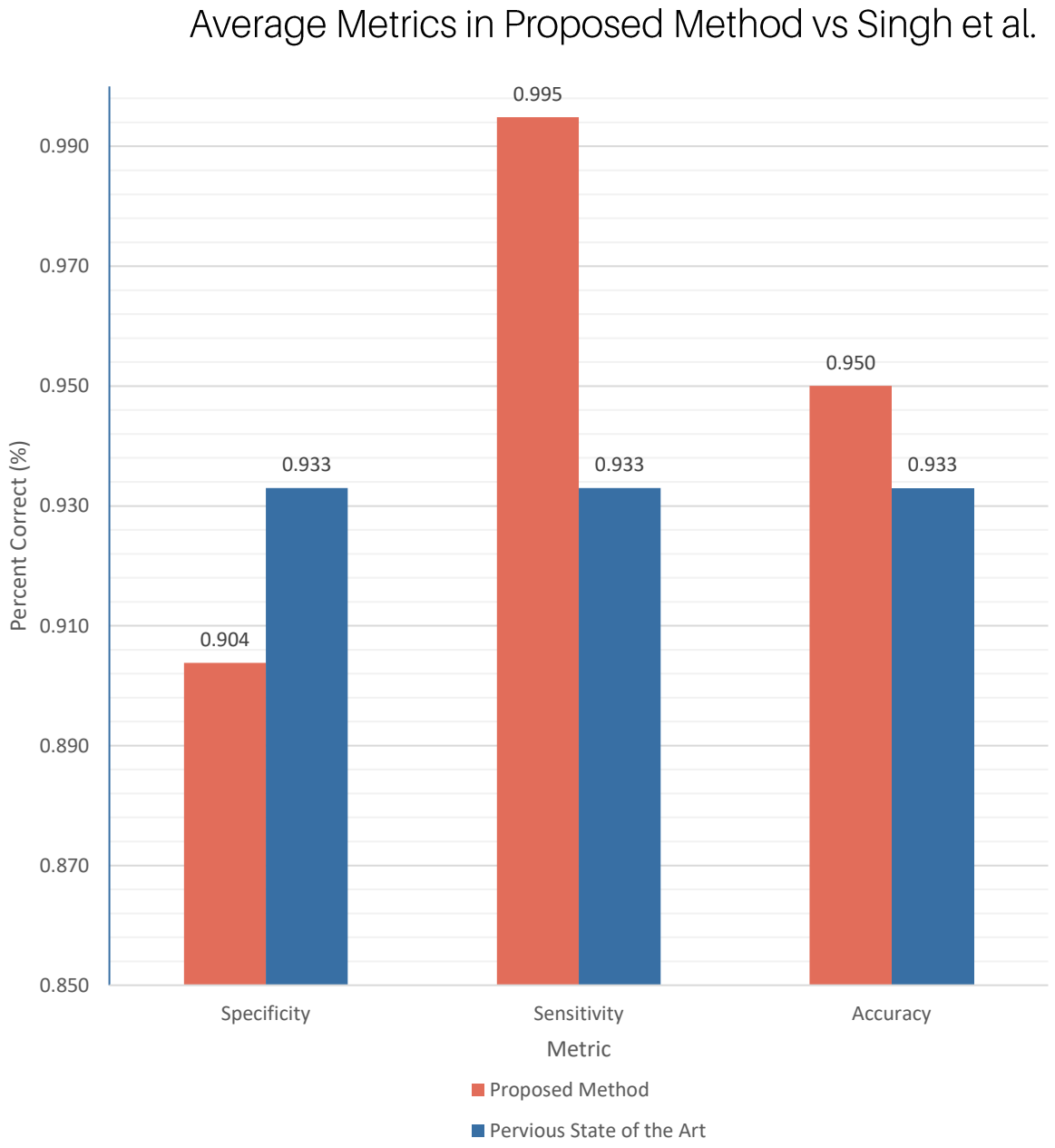
Sensitivity:

1.54E-77:

Accuracy

1.07E-11

From the p-values shown above, we can conclude that all metrics were statistically significant because all p-values were less than 0.05. This implies that the null hypothesis can be reject and alternative hypothesis be accepted. The decrease in specificity is expected because we prioritized sensitivity over specificity to maximize the true positive rate, the percent of correctly identified abnormal heart sounds from a sample of only abnormal heart sounds.

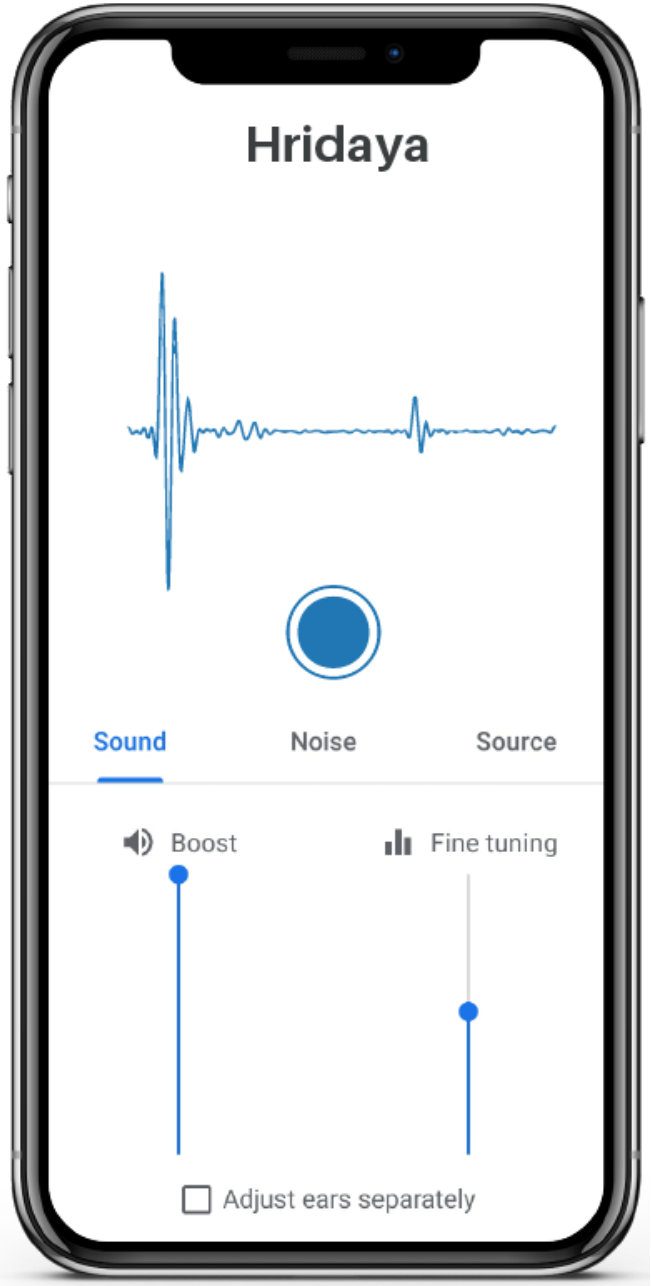


Discussion

We proposed a Generative Adversarial Network (GAN), composed of a Dense Generator and a Convolutional Neural Network (CNN) Discriminator to detect abnormal heart sounds in a recording. The best model achieved an accuracy of 98.75%, a specificity of 96.72%, and a sensitivity of 100% on the testing set. The previous state-of-the-art on the PASCAL dataset achieved an accuracy of 93.3%, a specificity of 93.3%, and a sensitivity of 93.3%. While the previous state-of-the-art on the 2016 PhysioNet CinC challenge dataset achieved an accuracy of 86.02 %, a specificity of 92.42%, and a sensitivity of 77.81%. This data, along with results from the t-test reveal that the proposed alternative hypothesis was correct and that the null hypothesis should be rejected. This is because the proposed method reached better performance than the previous state-of-the-art methods. Additionally, the model attained a staggering 1471 classification per second. This is because of the nature of the CNN architecture; unlike other methods, the CNN can forward propagate all inputs through the model. Architectures like RNNs and LSTMs require synchronous inputs, meaning the output of the model depends on its previous inputs; thus, the model must forward propagate through each element in the input. The object of this study was to create a fast and accurate model, capable of detecting arrhythmias in heart sounds. Thus, our proposed method not only accomplishes exemplary statistics but has the capabilities of doing so in real-time.

Further Exploration and Application

- Deploying the model with an app that is available to 3rd world countries that can't afford to conduct in-depth testing regularly
- Implementing datasets with a more variety of pathologies
  - Heart Sound and Murmur Library
- Further investigate feature extractions from PCG signals by visualizing the discriminator to discover new features for abnormal PCG detection
  - Class activation maps for CNNs
  - T-sne visualization
- Optimizing the time taken to detect an arrhythmia in a PCG
  - Allows for faster training and response times
- Recreate PCG signals based on the relationship between Electrocardiograms (ECGs).
  - The association between ECGs and PCGs is amended to translate from one space to another, where ECGs become dimensionally reduced, then constructed into a PCG signal.
  - Preexisting ECG datasets that contain a larger variety of arrhythmias can be recreated into PCG signals. These signals can be fed into a discriminator.
  - This proposed system makes use of existing ECG and transforms it into PCG data; hence, allowing for greater arrhythmia classification.



Thank You!

References

Abo-Zahhad, M., Farrag, M., Abbas, S. N., & Ahmed, S. M. (2016). A comparative approach between cepstral features for human authentication using heart sounds. *Signal, Image and Video Processing*, *10*(5), 843–851. https://doi.org/10.1007/s11760-015-0826-9

Ali, M. N., El-Dahshan, E.-S. A., & Yahia, A. H. (2017). Denoising of Heart Sound Signals Using Discrete Wavelet Transform. *Circuits, Systems, and Signal Processing*, *36*(11), 4482–4497. https://doi.org/10.1007/s00034-017-0524-7

Almasi, A., Shamsollahi, M. B., & Senhadji, L. (2011). A dynamical model for generating synthetic Phonocardiogram signals. *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 5686–5689. https://doi.org/10.1109/IEMBS.2011.6091376

Aziz, S., Khan, M. U., Alhaisoni, M., Akram, T., & Altaf, M. (2020). Phonocardiogram Signal Processing for Automatic Diagnosis of Congenital Heart Disorders through Fusion of Temporal and Cepstral Features. *Sensors*, *20*(13), 3790. https://doi.org/10.3390/s20133790

Babu, K. A., Ramkumar, B., & Manikandan, M. S. (2017). S1 and S2 heart sound segmentation using variational mode decomposition. *TENCON 2017 - 2017 IEEE Region 10 Conference*, 1629–1634. https://doi.org/10.1109/TENCON.2017.8228119

Bashar, Md. K., Dandapat, S., & Kumazawa, I. (2018). Heart Abnormality Classification Using Phonocardiogram (PCG) Signals. *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 336–340. https://doi.org/10.1109/IECBES.2018.8626627

References

Bau, D., Zhu, J.-Y., Strobelt, H., Zhou, B., Tenenbaum, J. B., Freeman, W. T., & Torralba, A. (2018). GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. *ArXiv:1811.10597 [Cs]*. http://arxiv.org/abs/1811.10597

D, P., T, U. M., K, P., & A, S. (n.d.). *Detection of Heart Diseases by Mathematical Artificial Intelligence Algorithm Using Phonocardiogram Signals*.

Dao, A. T. (2015). Wireless laptop-based phonocardiograph and diagnosis. *PeerJ*, *3*, e1178. https://doi.org/10.7717/peerj.1178

Das, A., Agrawal, H., Zitnick, C. L., Parikh, D., & Batra, D. (2016). Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions? *ArXiv:1606.03556 [Cs]*. http://arxiv.org/abs/1606.03556

Deperlioglu, O., Kose, U., Gupta, D., Khanna, A., & Sangaiah, A. K. (2020). Diagnosis of heart diseases by a secure Internet of Health Things system based on Autoencoder Deep Neural Network. *Computer Communications*, *162*, 31–50. https://doi.org/10.1016/j.comcom.2020.08.011

Electrocardiogram_generation_with_a_bidirectional_.pdf. (n.d.).

Fernando, T., Ghaemmaghami, H., Denman, S., Sridharan, S., Hussain, N., & Fookes, C. (2020). Heart Sound Segmentation using Bidirectional LSTMs with Attention. *IEEE Journal of Biomedical and Health Informatics*, *24*(6), 1601–1609. https://doi.org/10.1109/JBHI.2019.2949516

Finlay, M. (2006). The “mobile-phonocardiogram”, a new tool in the arrhythmia clinic. *Heart*, *92*(7), 898–898. https://doi.org/10.1136/hrt.2005.076315

Garg, V., Mathur, A., Mangla, N., & Rawat, A. S. (2019). Heart Rhythm Abnormality Detection from PCG Signal. *2019 Twelfth International Conference on Contemporary Computing (IC3)*, 1–5. https://doi.org/10.1109/IC3.2019.8844950

References

Ger, S., & Klabjan, D. (2020). Autoencoders and Generative Adversarial Networks for Imbalanced Sequence Classification. *ArXiv:1901.02514 [Cs, Stat]*. http://arxiv.org/abs/1901.02514

Ismail, S., Siddiqi, I., & Akram, U. (2018). Localization and classification of heart beats in phonocardiography signals—A comprehensive review. *EURASIP Journal on Advances in Signal Processing*, *2018*(1), 26. https://doi.org/10.1186/s13634-018-0545-9

Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., & Wu, Y. (2019). Direct speech-to-speech translation with a sequence-to-sequence model. *ArXiv:1904.06037 [Cs, Eess]*. http://arxiv.org/abs/1904.06037

Jordaens, L. (2018). A clinical approach to arrhythmias revisited in 2018: From ECG over noninvasive and invasive electrophysiology to advanced imaging. *Netherlands Heart Journal*, *26*(4), 182–189. https://doi.org/10.1007/s12471-018-1089-1

Kalaivani, V. (2014). DIAGNOSIS OF ARRHYTHMIA DISEASES USING HEART SOUNDS AND ECG SIGNALS. *Russian Journal of Cardiology*, *1-ENG*, 35–41. https://doi.org/10.15829/1560-4071-2014-1-ENG-35-41

Kiranyaz, S., Zabihi, M., Rad, A. B., Ince, T., Hamila, R., & Gabbouj, M. (2020). Real-time phonocardiogram anomaly detection by adaptive 1D Convolutional Neural Networks. *Neurocomputing*, *411*, 291–301. https://doi.org/10.1016/j.neucom.2020.05.063

Krishnan, P. T., Balasubramanian, P., & Umapathy, S. (2020). Automated heart sound classification system from unsegmented phonocardiogram (PCG) using deep neural network. *Physical and Engineering Sciences in Medicine*, *43*(2), 505–515. https://doi.org/10.1007/s13246-020-00851-w

References

Latif, S., Usman, M., Rana, R., & Qadir, J. (2020). Phonocardiographic Sensing using Deep Learning for Abnormal Heartbeat Detection. *ArXiv:1801.08322 [Cs]*. http://arxiv.org/abs/1801.08322

Liu, C., Springer, D., Li, Q., Moody, B., Juan, R. A., Chorro, F. J., Castells, F., Roig, J. M., Silva, I., Johnson, A. E. W., Syed, Z., Schmidt, S. E., Papadaniil, C. D., Hadjileontiadis, L., Naseri, H., Moukadem, A., Dieterlen, A., Brandt, C., Tang, H., … Clifford, G. D. (2016). An open access database for the evaluation of heart sound algorithms. *Physiological Measurement*, *37*(12), 2181–2213. https://doi.org/10.1088/0967-3334/37/12/2181

Lubaib, P., & Muneer, K. V. A. (2016). The Heart Defect Analysis Based on PCG Signals Using Pattern Recognition Techniques. *Procedia Technology*, *24*, 1024–1031. https://doi.org/10.1016/j.protcy.2016.05.225

Machine Learning Visualization. A collection of a few interesting… _ by Pier Paolo Ippolito _ Towards Data Science.pdf. (n.d.).

Messner, E., Zohrer, M., & Pernkopf, F. (2018). Heart Sound Segmentation—An Event Detection Approach Using Deep Recurrent Neural Networks. *IEEE Transactions on Biomedical Engineering*, *65*(9), 1964–1974. https://doi.org/10.1109/TBME.2018.2843258

Model Interpretability and Visualization_ Looking Inside the Neural Network Black Box _ by Avinash _ Heartbeat.pdf. (n.d.).

Olah, C., Mordvintsev, A., & Schubert, L. (2017). Feature Visualization. *Distill*, *2*(11), 10.23915/distill.00007. https://doi.org/10.23915/distill.00007

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. *ArXiv:1609.03499 [Cs]*. http://arxiv.org/abs/1609.03499

References

PhysioNet/CinC Challenge 2016: Training Sets. (n.d.). Retrieved January 11, 2021, from https://archive.physionet.org/pn3/challenge/2016/

Potes, C., Parvaneh, S., Rahman, A., & Conroy, B. (2016, September 14). *Ensemble of Feature:based and Deep learning:based Classifiers for Detection of Abnormal Heart Sounds*. 2016 Computing in Cardiology Conference. https://doi.org/10.22489/CinC.2016.182-399

Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., & Ng, A. Y. (2017). Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *ArXiv:1707.01836 [Cs]*. http://arxiv.org/abs/1707.01836

Rawther, N. N., & Cheriyan, J. (2015). *Detection and Classification of Cardiac Arrhythmias based on ECG and PCG using Temporal and Wavelet features*. *4*(4), 6.

S1 and S2 Heart Sound Recognition Using Deep Neural Networks. (2017). *IEEE Transactions on Biomedical Engineering*, *64*(2), 372–380. https://doi.org/10.1109/TBME.2016.2559800

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). SmoothGrad: Removing noise by adding noise. *ArXiv:1706.03825 [Cs, Stat]*. http://arxiv.org/abs/1706.03825

Subasi, A. (2019). Biomedical Signals. In *Practical Guide for Biomedical Signals Analysis Using Machine Learning Techniques* (pp. 27–87). Elsevier. https://doi.org/10.1016/B978-0-12-817444-9.00002-7

Surukutla, D., Bhanushali, K., & Patil, T. (2020). Cardiac Arrhythmia Detection Using CNN. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3572237

References

Thoms, L.-J., Collichia, G., & Girwidz, R. (2019). Real-life physics: Phonocardiography, electrocardiography, and audiometry with a smartphone. *Journal of Physics: Conference Series*, *1223*, 012007. https://doi.org/10.1088/1742-6596/1223/1/012007

Yupapin, P., Wardkein, Yupapin, P., Phanphaisarn, Koseeyaporn, Roeksabutr, Roeksabutr, Wardkein, & Koseeyapon. (2011). Heart detection and diagnosis based on ECG and EPCG relationships. *Medical Devices: Evidence and Research*, 133. https://doi.org/10.2147/MDER.S23324

Zeiler, M. D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. *ArXiv:1311.2901 [Cs]*. http://arxiv.org/abs/1311.2901

Zhang, W., Han, J., & Deng, S. (2017). Heart sound classification based on scaled spectrogram and partial least squares regression. *Biomedical Signal Processing and Control*, *32*, 20–28. https://doi.org/10.1016/j.bspc.2016.10.004

Zhao, C., Sun, Q., Zhang, C., Tang, Y., & Qian, F. (2020). Monocular Depth Estimation Based On Deep Learning: An Overview. *Science China Technological Sciences*, *63*(9), 1612–1627. https://doi.org/10.1007/s11431-020-1582-8

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning Deep Features for Discriminative Localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2921–2929. https://doi.org/10.1109/CVPR.2016.319