1

Cardiac Cadence: Leveraging ML for Heartbeat Classification

Alex Atherton and Hadassah Lurbur ara3@williams.edu; hnl2@williams.edu
Williams College Computer Science

1. Introduction

Cardiovascular diseases are the leading cause of death worldwide, representing approximately 32% of global deaths each year. Additionally, over 75% of deaths from cardiovascular diseases take place in low and middle-income countries W.H.O (2021). Typically, diagnosis of heart conditions has relied on in-person evaluation of patients. We aim to tackle this issue and provide accessible heartbeat evaluation by implementing a machine-learning model capable of classifying heartbeat sounds into disease categories. More specifically, convert existing heartbeat audio into spectrogram images to then train a convolutional neural network to accurately classify patient heartbeat audio. Additionally, because we have a relatively small dataset of only 661 audio files we implemented a baseline random forest model to combat the variance problem introduced by our small dataset size. Existing literature on heartbeat classification has primarily focused on using ECG (Electrocardiogram) data for classification. While in-person EEG analysis produces highly accurate results, this method is not widely accessible. While our model produces slightly less accurate results, it provides a widely accessible method for heartbeat analysis. This could allow people without easy access to healthcare to get an early diagnosis of heart conditions and hopefully decrease deaths from cardiovascular diseases.

2. Preliminaries

Audio classification tasks typically rely on the conversion of audio files to spectrograms, which are visual representations of the signal strength over time at various frequencies present in a particular waveform. Audio signals are comprised of several single-frequency sound waves; the Fourier transform Fourier (1822) allows us to decompose a signal into its frequencies and their respective amplitude, converting the signal from a time domain to a frequency domain. See Figure 1 for an example of the Fourier transform applied to an audio signal.

To generate a spectrogram we apply a multitude of overlapping Fourier transforms to the audio file and stack them on top of each other. The resulting spectrogram then takes the form of an image akin to Figure 2. Digital images are stored as matrices of numbers where each entry in the matrix specifies the brightness of the pixel. Specifically, RBG images, have a matrix corresponding to each of their color channels, red, blue, and green.

First, we deployed a random forest model to establish baseline classification accuracy. Random forests are a relatively simple model, yet are still capable of performing classification on images. Additionally, a random forest is a fitting baseline for this project given that they are a bagged algorithm that will mitigate variance introduced by our relatively small dataset.

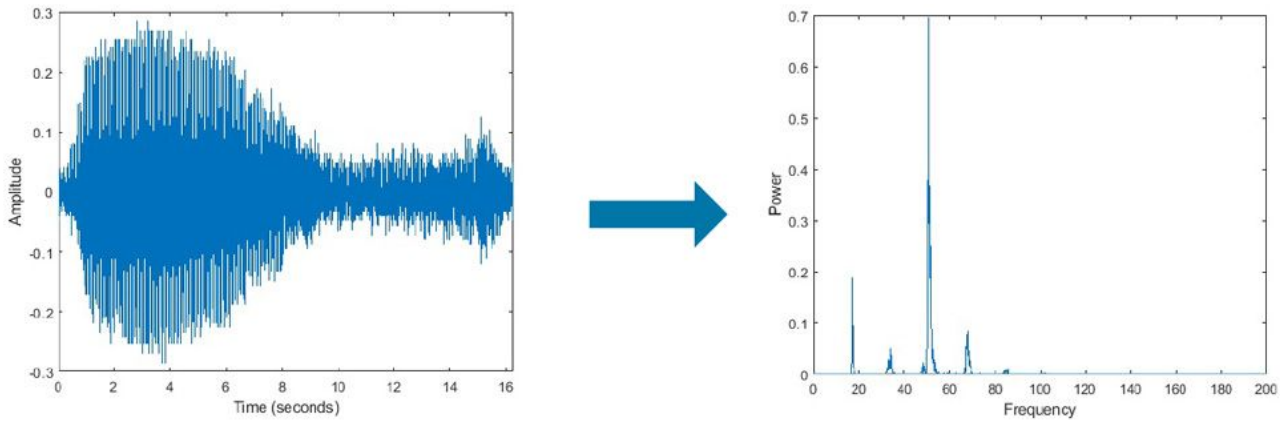

Figure 1: Application of Fourier Transform

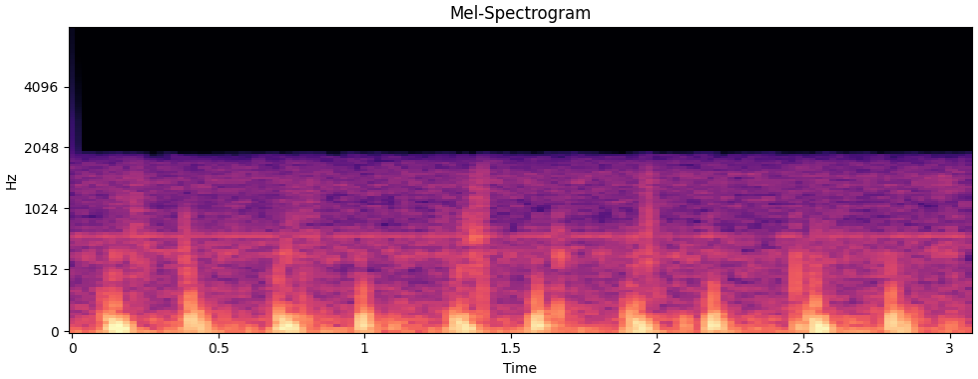

Figure 2: Normal Heartbeat Example Spectrogram

A random forest is a supervised learning model that produces results by averaging the output of many decision trees. Within the random forest, each decision tree produces a classification result by moving a data point down the splits of the tree from the root to a node. The structure of the tree is learned by minimizing the entropy of the dataset at each split, given by the following formula where H is entropy, and p_c denotes the proportion of data points in [D] where Y has the class label c.

$$H([D]) = -\sum_{c} p_c \log_2(p_c)$$

In a decision tree of unlimited depth, the model will continue to split the data until the entropy of each node is 0 and the data has been perfectly split. Because trees of unlimited depth overfit the data, they are high-variance models. Utilizing the individual high variance of unlimited depth decision trees, random forests employ bagging by averaging the results of many high variance decision trees. Created by Breiman (2001) random forests are made by sampling m datasets $[D]_1 \ldots [D]m$ of size n from [D] without replacement. For each $[D]_i$ a full decision tree $h[D]_i$ of unlimited depth is trained where each split in $h[D]_i$ is made on $k \leq d$ randomly subsampled features. The final aggregated random forest model is the mean or mode of m individual decision trees as is done in regular bagging.

Next, we deploy a CNN (Convolutional Neural Network) Lecun et al. (1998) as our machine learning model. CNNs consist of four layer types: a convolutional layer, an activation layer, a pooling layer, and

a fully connected layer. In the convolutional layer, we pass kernels over the input image to produce a feature map. The feature map is the output of the convolution layer and indicates the presence or absence of specific features in the image. Each kernel is a matrix of values that transforms the image based on its values as it passes over the image. Each kernel is responsible for detecting certain features in the image. Applying the kernel to the image involves computing the dot product of the kernel and the image. The feature map is the collection of dot products computed as kernels are passed over the image and is the output of the convolution layer.

In the activation layer, an activation function is applied which introduces non-linear complexity to the model, allowing the model to learn complex features from the input image. Common activation functions include ReLU and sigmoid. We chose to use ReLU in our model due to its relatively low computational cost.

Next, the feature map output by the convolution layer is passed to the pooling layer. The pooling layer sweeps a kernel across images in a similar manner to the convolution layer but reduces the number of parameters from the input. This downsizing of parameters reduces computational complexity and increases the CNN's computational efficiency. The outputs of the pooling layer are then passed into another convolutional layer which is then fed into another pooling layer, and finally into a fully connected layer, where image classification takes place.

In the fully connected layer, the output of the prior layers is used to classify each image as either normal, artifact, murmur, extrahls, or extrastole. This classification is the final output of the CNN.

Finally, we use K-fold cross validation to evaluate the performance of our CNN and select the optimal model for deployment on the test set. This involves splitting the training data in to K subsets, or folds. Then the model is trained and evaluated K times. For each evaluation, it trains on K-1 folds and is evaluated on the remaining fold. This process allows for better evaluation of the models performance than a single train-test split, and allows us to better utilize our small data set to optimize our model. We chose a K value of 10, which is broadly used K value for K-fold cross validation for machine learning sklearn (2023).

3. Data

Our data is a set of heartbeat audio recordings, sourced from the general public as well as hospital clinical trials. Data recorded by the public was recorded using the iStethoscope Pro iPhone app while hospital-collected data was recorded using a digital stethoscope called DigiScope. All recordings in the dataset have class labels for classification. The classes in the dataset are as follows: normal, artifact, murmur, extrahls, and extrastole. The audio files are of varying lengths; the longest recording is 120 seconds, the shortest recording is 1 second, and the median recording length is 11 seconds. The audio recordings are clean; they only contain noise from the heartbeat and do not have any background noise.

Our data preprocessing can be segmented into multiple steps. All of the audio files were initially stored as .wav files. The first step was to convert every .wav file into a spectrogram that is stored as a .png file, where the file ID and class were contained in the filename. We then created a Huggingface dataset with predefined train test splits from the folder, such that each entry in the dataset has a spectrogram that is a Python image library(PIL) image and a class label. The train split is 80 percent of the original audio data (528 entries), while the test set is 20 percent (133 entries). Our motivations behind making a Huggingface dataset were as follows: 1. Storing the images in the cloud would alleviate having to store hundreds of images on disk. 2. Creating a spectrogram is relatively expensive, storing the spectrograms

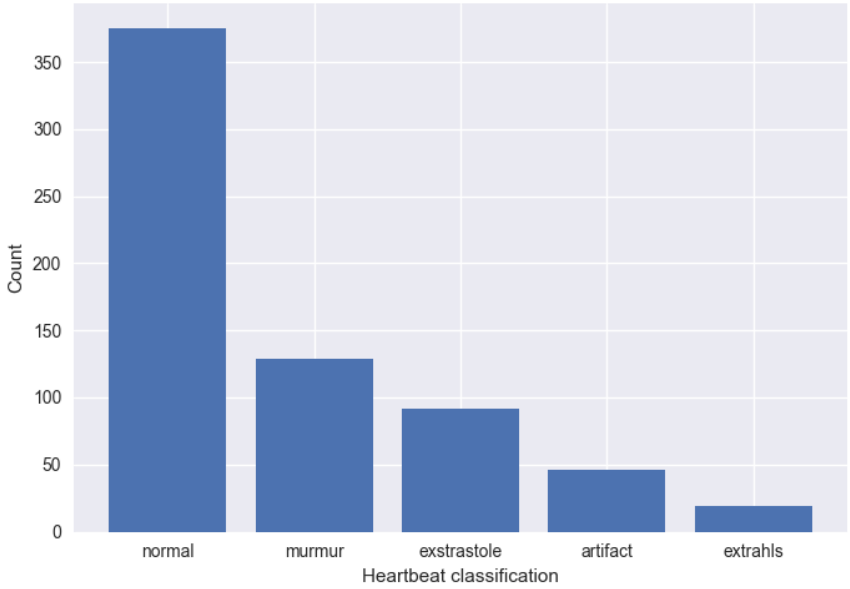

Figure 3: Class label distribution

in a dataset avoids repeating this work. 3. Creating a public access dataset will allow others to build off of our work without needing to convert the audio files to spectrograms.

Data preprocessing for random forests vs. CNN model training was slightly different as our random forest class uses sklearn while the CNN model uses pytorch. The PIL images are first converted from RGBA format to RGB format; the alpha channel gets dropped as opacity is irrelevant in the spectrogram. Since some of the images are different sizes, the next step is to resize them to a common height and width. We then convert the transformed PIL images into either numpy arrays or tensors depending on whether we are training the random forest algorithm or a CNN. If training the random forest algorithm the data is then formatted into a pandas dataframe, if training a CNN the data is then formatted into pytorch dataloaders.

4. Training And Validation Of Models

We began by training five random forest models as baseline classifiers. The random forests all had forest sizes of 100 and max depths of 1, 5, 10, 25, and 50, respectively.

For our CNN model, we used a 10-fold cross-validation to train the model. On each fold we trained the model for 50 epochs, recording accuracy scores on the validation set every five epochs. We chose to use cross-validation as opposed to a validation set due to the already small size of our dataset. The accuracy scores can be seen in Figure 4.

We can see that across the folds, the models tend to perform worse in terms of accuracy on their cross-validation sets after 40 training epochs. Due to the skewed class distribution discussed in Section 2, accuracy is an imperfect metric to measure model performance—a model that always classifies an image as "normal" could perform relatively well in terms of accuracy, while not being informative for classification

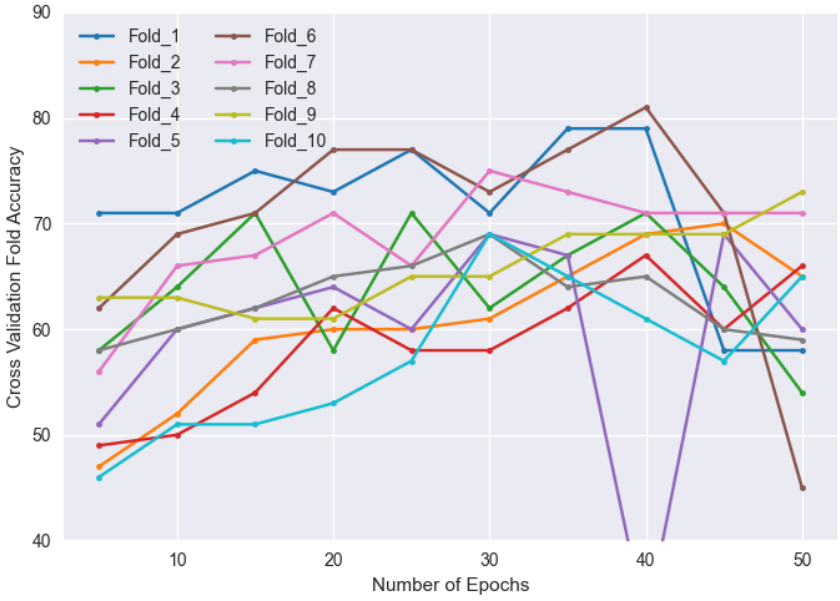

Figure 4: Cross-Validation Performance

tasks. To get a better sense of model performance, we generated confusion matrices for each of our models.

Models also begin by always predicting normal and eventually the models spread out their guess distribution but have lower accuracy as they often misclassify normal heartbeat audio as other types. See Figures 5 and 6 for visualizations of these poorly performing model confusion matrices.

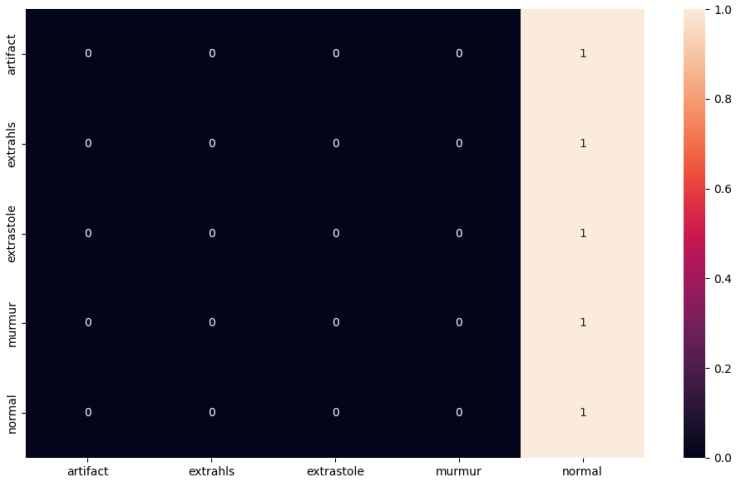

Figure 5: Fold 6 Checkpoint 5

Two models from the first fold had accuracy scores of 79.0 and the best-performing model from the sixth fold had an accuracy score of 81.0. To select our final model, we chose to evaluate the confusion

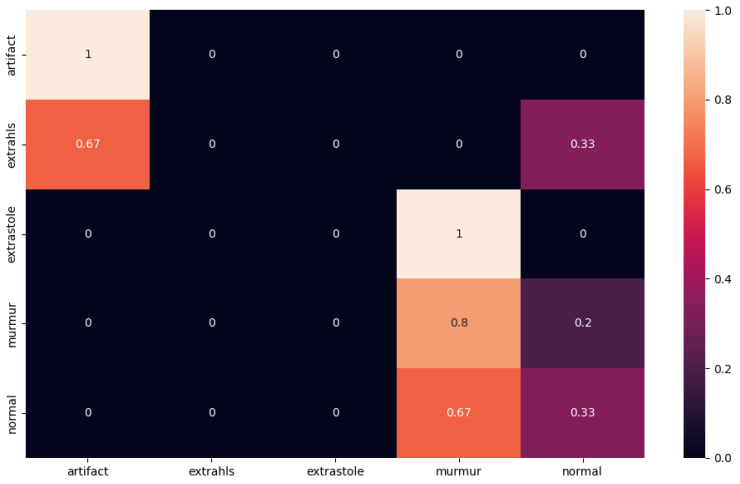

Figure 6: Fold 6 Checkpoint 50

matrices for each of these. See Figures 7, 8, and 9 below for a visualization. Investigation of other relatively high-performing models deemed them as noncandidates for generalization as their relatively high accuracy scores could be attributed to naively classifying nearly all data as normal.

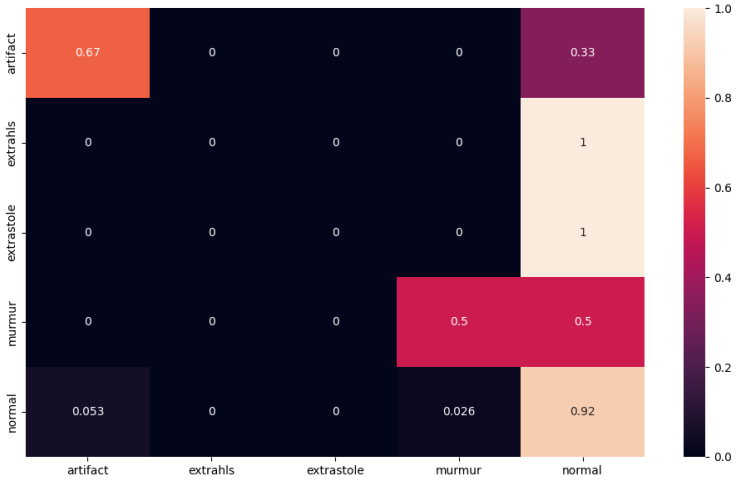

Figure 7: Fold 1 Checkpoint 35

The model from the 40th training epoch of the sixth fold seems to perform the best across classes of these three models. None of them were able to correctly predict extrahls or extrasole on their cross-validation sets. The aforementioned model was better at classifying murmur and artifact classes on its cross-validation set, making it seem like the most appropriate choice to advance to testing as it seemed most capable of generalizing to unseen data.

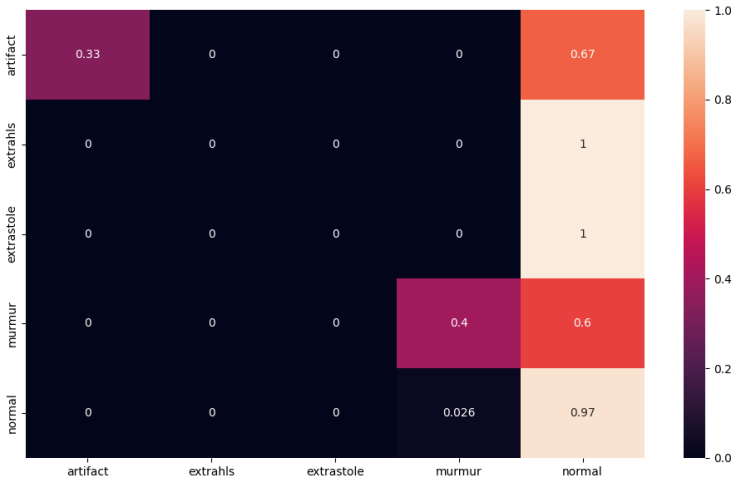

Figure 8: Fold 1 Checkpoint 40

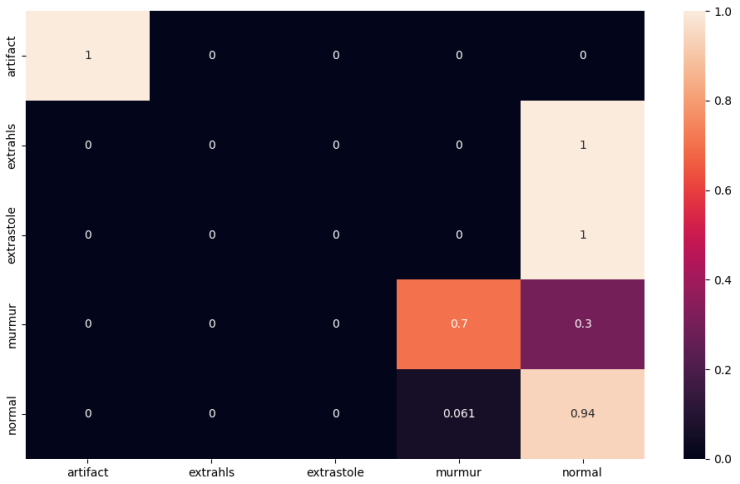

Figure 9: Fold 6 Checkpoint 40

We concluded the training and validation phase by fine-tuning this model on the remainder of the training data that it had not been trained on due to the nature of cross-validation.

5. Results

We then tested our 5 Random Forest classifiers to establish the highest baseline possible as well as our finetuned CNN from fold 6 checkpoint 40 from the training phase. Accuracy scores on the test data can be seen in Table 1.

Based on these results, the CNN was able to outperform all of our baseline models in terms of raw accuracy on the test data. We then wanted to get a feel for how the CNN compared to the highest-

Model	Accuracy
$RFDepth_1$	0.65
$RFDepth_5$	0.68
$RFDepth_{10}$	0.69
$RFDepth_{25}$	0.70
$RFDepth_{50}$	0.69
$CNN_{fold_6 epoch_{40}}$	0.72

Table 1: Accuracy of Random Forest Classifier at different depths

performing baseline model within each class. The random forest of depth 25 has a confusion matrix that can be seen in Figure 10. The CNN model has a confusion matrix that can be seen in Figure 11.

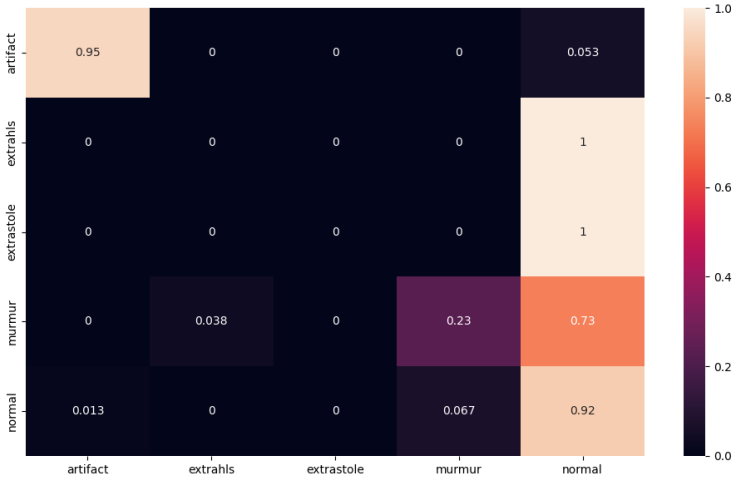

Figure 10: Random Forest Depth 25 Confusion Matrix

The Random Forest Model does a good job of classifying artifact and normal heartbeat audio on the test data, correctly classifying them 95 and 92 percent of the time, respectively. The model does a poor job of classifying murmur, extrastole, and extrahls heartbeat audio on the test set. It always classified extrastole and extrahls as normal, and only correctly classified murmur 23 percent of the time. This suggests that our baseline random forest model is heavily prone to type 2 errors, often (or always) misclassifying negative heartbeat conditions as normal.

The CNN model also does a good job classifying artifact and normal heartbeat audio on the test data, classifying them correctly 79 and 92 percent of the time, respectively. The artifact performance is significantly worse than the baseline model. The model was better at classifying murmur, correctly classifying it 46 percent of the time versus 23 percent of the time.

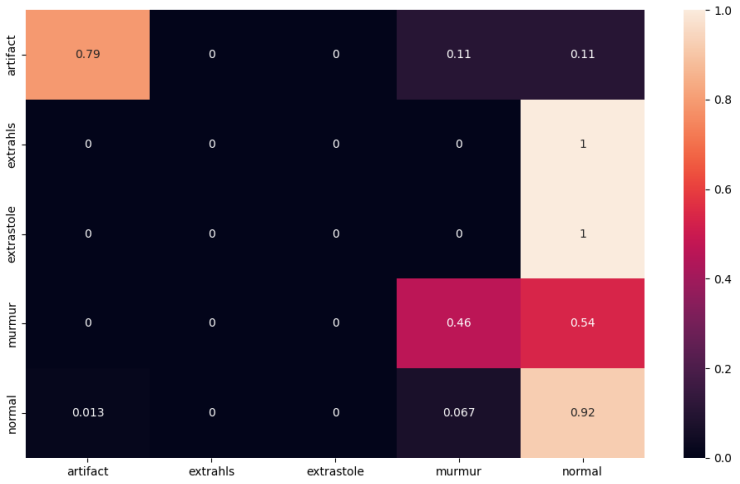

Figure 11: CNN Confusion Matrix

6. Ablation Study

For our ablation study, we introduced noise by applying a Gaussian Blur to the images in the test set. The spectrogram in Figure 2 takes the form of Figure 12 after applying a Gaussian blur.

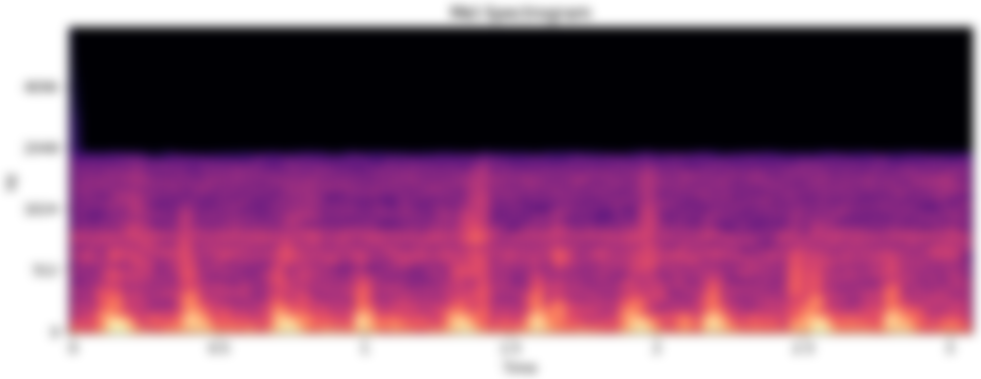

Figure 12: Example Spectrogram Gaussian Blur

Our CNN model reported an accuracy of 66.0 with this noise added to the test set. The random forest reported an accuracy of 66.2. The confusion matrix for the CNN model is seen in Figure 13, and the confusion matrix for the random forest is seen in Figure 14.

The random forest we have been using as a baseline classifier was actually better at classifying the noisy test data than the CNN model. This is seen by the marginally higher accuracy score. Notably, it was much better at classifying artifact and murmur (95 and 31 percent) as compared to the CNN (68 and 19 percent). The CNN was much better at classifying normal (95 vs. 81 percent).

Takeaways from this are somewhat limited. A Gaussian Blur is likely not fully reflective of noise that actually represents variance in unseen data. Other forms of noise that could exist could be background

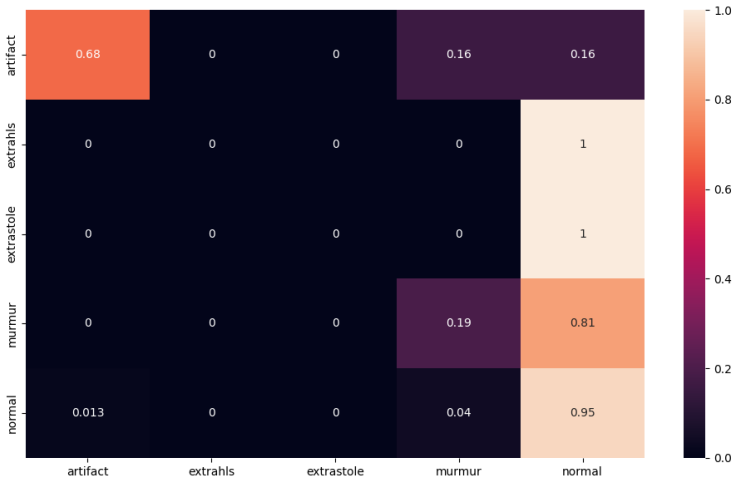

Figure 13: CNN Confusion Matrix - Noisy Data

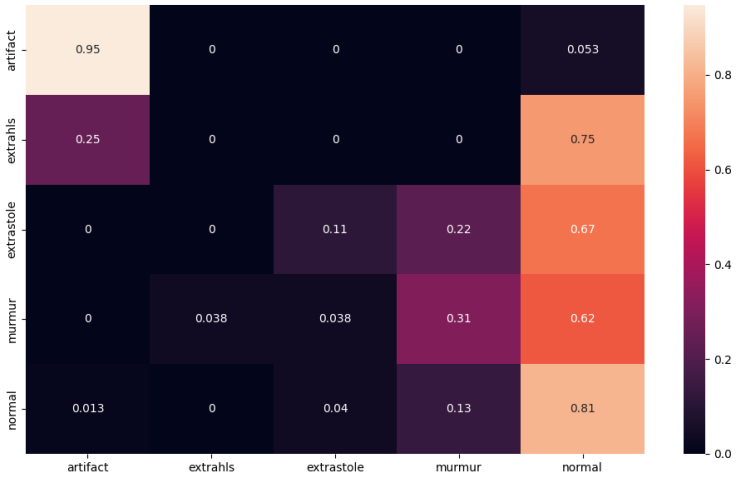

Figure 14: Random Forest Depth 25 Confusion Matrix - Noisy Data

noise, insufficiently short audio recordings, or low-quality recordings. Thus, blurring the images could be considered an imperfect way to measure the model's ability to generalize. Nevertheless, it is disheartening that the CNN seemingly learned less about the shape patterns of spectrograms of these classes than we would have anticipated. Interestingly, the baseline model did a better job of predicting murmur than it did on non-noisy data and was worse at predicting normal. This can possibly be attributed to the two being visually similar and to some degree shows the unsuitability of the baseline model. Further work could expand on this ablation study by implementing more nuanced versions of noise that replicate real-world concerns about the extensibility of our model.

7. Discussion and Conclusion

Overall, our accuracy of approximately 70% and 72% for the baseline random forest model and CNN falls short of our goals for this project. We hypothesize that this relatively low accuracy is rooted in a shortage of data for abnormal heart sounds. In both models, we observed high type II error rates where the model falsely classified abnormal sounds as normal. Our training data has very few samples of abnormal heart sounds, likely causing the model to overfit to these samples. This leads to high error rates for abnormal sounds which have very few samples in the data, resulting in high type II error.

While the CNN only achieved marginally higher accuracy over our random forest, it did have lower rates of type two error. The CNN performed better than the random forest when classifying categories with relatively few data points. We hypothesize this is due to the CNN's ability to perform more complex features within the image using convolution layers.

Because the type II errors we observed were higher for categories with fewer data points in the training data, we feel that these error rates would improve if we obtained more data for abnormal heart sounds. Additional samples of abnormal sound would likely increase the accuracy of both the random forest and CNN. Additionally, we expect that due to the added complexity of the CNN, the CNN should perform better relative to the random forest with additional data.

We do not put too much weight on the results of the ablation study for the reasons outlined above in section 6. Nevertheless, the poor performance, especially related to our CNN being even more prone to type II error reflects poorly on the suitability of our model.

Overall, our model does not perform accurately enough to deploy in a clinical setting, especially given the high rate of type II errors which we specifically aim to minimize.

References

Leo Breiman. *Random Forests*. Springer Netherlands, 2001.

Jean Baptiste Joseph Fourier. *Théorie Analytique de la Chaleur*. Cambridge Library Collection - Mathematics. Cambridge University Press, 1822. doi: 10.1017/CBO9780511693229.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

sklearn. Cross validation: Evaluating estimator performance, 2023.

W.H.O. Cardio vascular diseases, 2021. `https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)` [Accessed: 11/28/23].