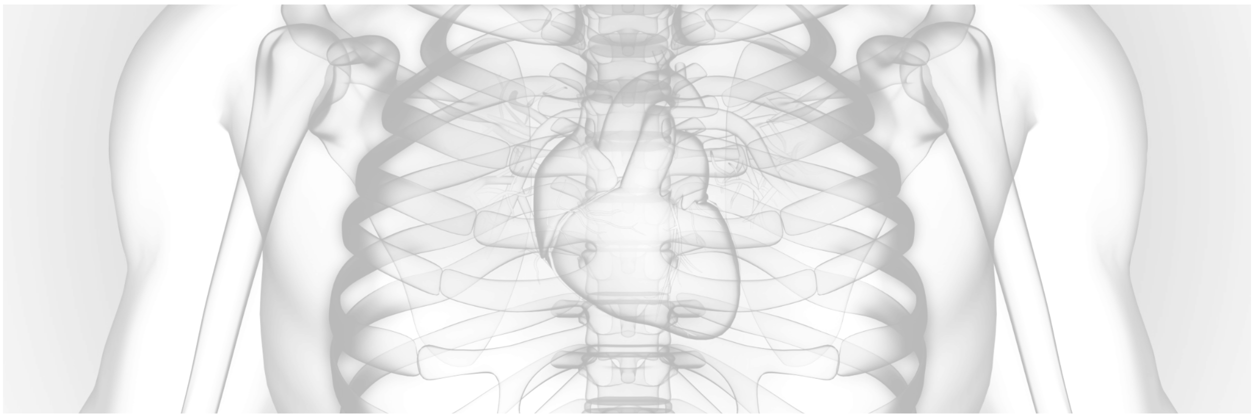
# 50.039 Theory and Practice of Deep Learning

Report

## Topic: **Classifying Heartbeat Anomalies from Audio**

| | |
|---|---|
| 1004662 | Ramya Sriram |
| 1004665 | Tan Yan Siew |
| 1003625 | Wong Kai Kang |
| 1004521 | Xu Song |
| 1003886 | Alyssa Ng |
| 1002994 | Zhu Dan |



# Content

## Topic

Our project aims to use the most suitable deep learning model to classify heartbeat anomalies from stethoscope audio.

## Problem

Generally, heart beats are monitored by doctors listening through a stethoscope. This is an important step in identifying potential heart anomalies that may need further health investigation.

However, heart beat sound classification is challenging due to the complex nature of audio data, frequency domains and its dynamic properties of time.  It is difficult to detect heart conditions with short data samples or unbalanced and noisy data.

Therefore, our group aims to produce a model that can classify real heart beat sounds as one of three possible labels (normal, murmur, extrahls).

## Dataset

The dataset chosen, Heartbeat Sounds, was retrieved from Kaggle, https://www.kaggle.com/datasets/kinguistics/heartbeat-sounds. This dataset was put together by Ed King.

The dataset includes 2 .csv files and over 600 audio files. The dataset is split into 2 different sets:

Set_a contains data collected from the general public through an *iPhone Application*.
Set_b contains data collected from clinical trials using a *Digital Stethoscope.*

| Dataset | |
|---|---|
| set_a.csv | Labels and metadata for heart beats collected from the general public via an **iPhone app** |
| set_a folder | Folder containing audio clips from **set_a** |
| set_a_timing.csv | Contains gold-standard timing information for the "normal" recordings from Set A |
| **set_b.csv** | Labels and metadata for heart beats collected from a clinical trial in hospitals using a **digital stethoscope** |
| **set_b folder** | Folder containing audio clips from **set_b** |

As our project aims to classify based on the audio sound and not the way it was collected, the method should be uniformed so as to not add another complex variable. Hence, we are working with set_b, as the data was obtained in a professional setting with more precise instruments. A higher standard of data will give us more precision with our models.

Set_b has 656 data samples for us to work with, which we hope to be sufficient to train and test our model.

The table belows shows the types of audio and its corresponding number of audio files:

| Audio type | # instances |
|---|---|
| Normal | 200 |
| Noisy normal | 120 |
| Normal total | 320 |
| Murmur | 66 |
| Noisy murmur | 29 |
| Murmur total | 95 |
| Extrastole | 46 |
| Abnormal total | 141 |

| Training total | 461 |
|---|---|
| Testing total (unlabelled) | 195 |
| Noisy total | 149 |

The “testing total” outputs are undefined, making it difficult to test our model and get an accuracy score. Hence, we can split the training data into 80% training, 10% validation and 10% testing data. The more data we train, the more accurate our model will likely be. With this method, we can maximize the amount of data trained and still test our method on a new batch of data.

## Inputs and Outputs of Trained Model

|  |  |
|---|---|
| **Input** | **Audio Clips**<br>● .wav file type<br>    ○ Varying lengths, between 1 second and 30 seconds.<br>**CSV Files**<br>● Includes all audio files and its corresponding class label (i.e. normal, murmur, extrahls)<br>● Unlabelled testing data |
| **Output** | **Class labels for audio files used as testing data.**<br>There are 5 possible outputs for the class labels in the whole dataset, while only 3 possible outputs for the class labels in set_b: normal, murmur, extrashole (if we are going to use only set_b).<br><br><For merged set_a and set_b><br>● "normal"<br>● “murmur”<br>● “extrahls”<br>● “artifact”<br>● “extrastole”<br><br><For set_b only><br>● "normal"<br>● “murmur”<br>● “extrastole” |

## Architecture

To classify the audio files based on their class labels, we had run the files through three stages: Feature Extraction, Data Preprocessing, and LSTM Model.

### Feature Extraction

There are a few more ways in which audio data can be represented:

- Time Domain features (eg. RMSE of waveform)
- Frequency domain features (eg. Amplitude of individual frequencies)
- Perceptual features (eg. MFCC)
- Windowing features (eg. Hamming distances of windows)

Out of the total of 656 audio files, we narrowed to only use the labeled data because the unlabeled data would neither help us train nor test the model. For the feature extraction, we chose to use Mel-frequency cepstrum coefficients as it is the most successful speech processing used. We have to transform the audio into meaningful tensors that we can use as input for neural networks. MFCC works by first doing a fourier transformation, then doing a cos function on a mel scale.

Mel Frequency Cepstral Coefficient (MFCC) is by far the most successful feature used in the field of Speech Processing. Speech is a non-stationary signal. As such, normal signal processing techniques cannot be directly applied to it.

### Data Preprocessing

For data preprocessing, we would need to manipulate our audio files as well as our csv files. We use the librosa package that is dedicated for music and audio analysis to load our .wav audio for feature extracting that will happen afterwards. For “set_a.csv” and “set_b.csv” that contain information like filename, labels, and sub labels, as files on set_a and set_b are named in different manners, we would then need to manipulate the filenames such that they are standardized. What is more, as we are only interested in the labeled data, unlabeled audio files will be dropped out. Finally, we write the labeled filenames and corresponding labels of data from set a and set b to a new csv file “MainData.csv” such that we can visualize and have a better understanding of our updated filenames and indices of files that will be used for training and testing later.

### Model Used

For audio classification, both Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) are suitable. However, LSTM has an advantage over RNN. While RNNs can only remember short-term information, LSTM are gated RNNs that can dynamically remember long-term and short-term information. Moreover, RNN faces the Vanishing Gradient Problem, an unstable behavior of a multilayer neural network, while LSTM has a better gradient flow. Thus, we have chosen to move forward with the LSTM as our classification model.

In the model, we chose to use softmax as our activation function in the output layer of the neural networks as we are doing multi-class classification for this project. Softmax gives a probability distribution. For each node in the output layer, softmax outputs its probability with a total sum of 1. Thus, the output label would be the one with the highest probability.

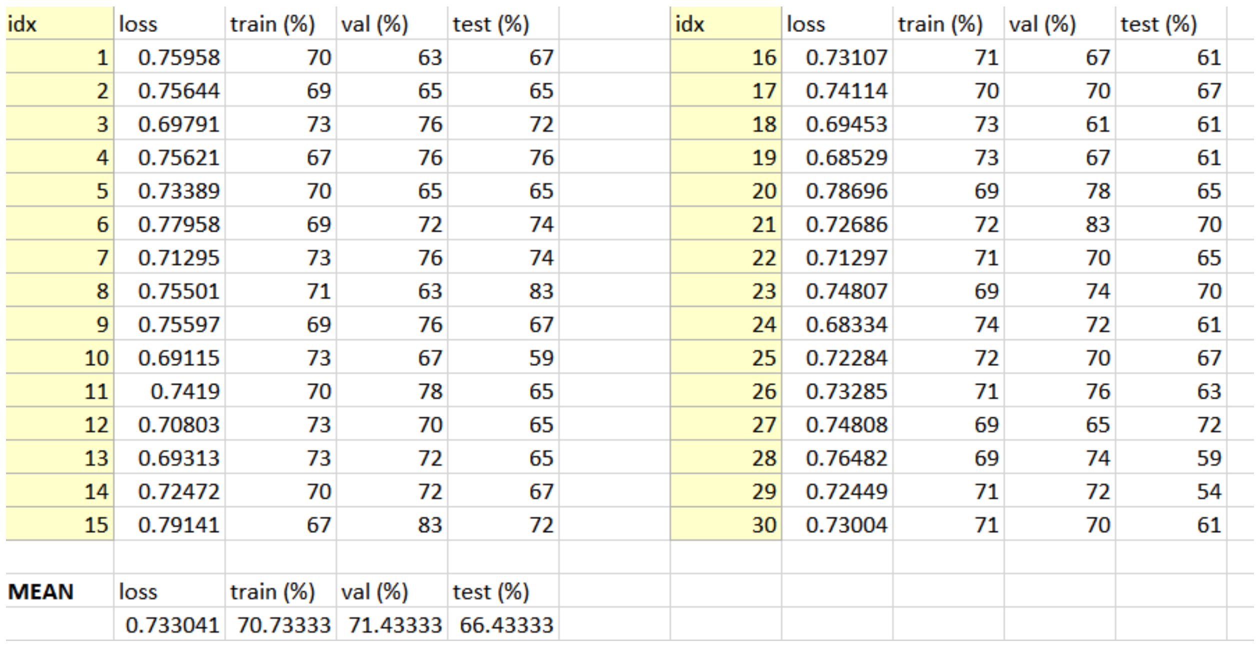
We chose AdaMax as our optimization function, which can be considered as an extension to the Gradient Descent. As for the keras metrics we used to evaluate our model’s performance, our choices are a few simple metrics including accuracy, mean square error, mean absolute error, mean absolute percentage error, and cosine proximity.

## Analysis

In order to train our LSTM model, we shuffled and  divided our labeled dataset as follows: 80% as training set, 10% as validation set, and 10% as testing set. As we are using set_b data, among the total 461 labeled audio data, we used 369 for training, 46 for validation, and the remaining 46 for testing.

As shown in the table below, as our dataset size is rather small, our model's performance is greatly affected by the results of the dataset splitting. For instance, the test accuracy ranges from 54% to 83%. As the testing dataset includes only 46 samples, one different classification would result in more than 2% change in test set accuracy.

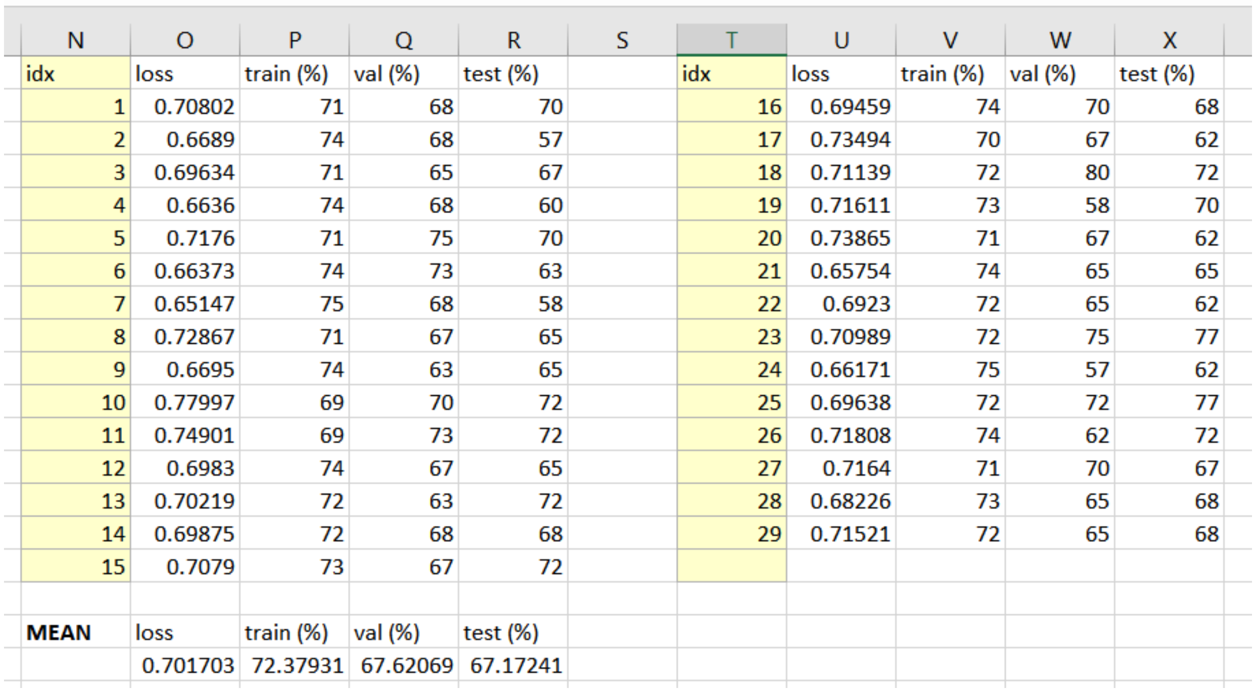
In order to reduce such impact of randomness on our model evaluations, we reshuffled the dataset and repeated the training and testing process for 30 times and calculated the mean loss and mean accuracy such that the calculated values would be better representations of the model's performance. We obtained an average loss of 0.733 with 66.4% accuracy on the test data on average.



| idx | loss | train (%) | val (%) | test (%) |
|---|---|---|---|---|
| 1 | 0.75958 | 70 | 63 | 67 |
| 2 | 0.75644 | 69 | 65 | 65 |
| 3 | 0.69791 | 73 | 76 | 72 |
| 4 | 0.75621 | 67 | 76 | 76 |
| 5 | 0.73389 | 70 | 65 | 65 |
| 6 | 0.77958 | 69 | 72 | 74 |
| 7 | 0.71295 | 73 | 76 | 74 |
| 8 | 0.75501 | 71 | 63 | 83 |
| 9 | 0.75597 | 69 | 76 | 67 |
| 10 | 0.69115 | 73 | 67 | 59 |
| 11 | 0.7419 | 70 | 78 | 65 |
| 12 | 0.70803 | 73 | 70 | 65 |
| 13 | 0.69313 | 73 | 72 | 65 |
| 14 | 0.72472 | 70 | 72 | 67 |
| 15 | 0.79141 | 67 | 83 | 72 |
| 16 | 0.73107 | 71 | 67 | 61 |
| 17 | 0.74114 | 70 | 70 | 67 |
| 18 | 0.69453 | 73 | 61 | 61 |
| 19 | 0.68529 | 73 | 67 | 61 |
| 20 | 0.78696 | 69 | 78 | 65 |
| 21 | 0.72686 | 72 | 83 | 70 |
| 22 | 0.71297 | 71 | 70 | 65 |
| 23 | 0.74807 | 69 | 74 | 70 |
| 24 | 0.68334 | 74 | 72 | 61 |
| 25 | 0.72284 | 72 | 70 | 67 |
| 26 | 0.73285 | 71 | 76 | 63 |
| 27 | 0.74808 | 69 | 65 | 72 |
| 28 | 0.76482 | 69 | 74 | 59 |
| 29 | 0.72449 | 71 | 72 | 54 |
| 30 | 0.73004 | 71 | 70 | 61 |

| **MEAN** | loss | train (%) | val (%) | test (%) |
|---|---|---|---|---|
|  | 0.733041 | 70.73333 | 71.43333 | 66.43333 |

As we felt that set_b alone was insufficient to generate satisfying results, we decided to enlarge our dataset by adding labeled audio files from set_a and merging them with the samples we have in set_b. This helped us to increase our total number of labeled sample to 585, among which we randomly picked 465 instances as our training samples, and 60 instances each as our validation and testing samples. Similarly, by repeating the shuffling-training-testing procedure for 29 times, we obtained the table below. Our average loss decreased from 0.733 to 0.701, while our test set accuracy increased from 66.4% to 67.2%.



| N | O | P | Q | R | S | T | U | V | W | X |
|---|---|---|---|---|---|---|---|---|---|---|
| idx | loss | train (%) | val (%) | test (%) | | idx | loss | train (%) | val (%) | test (%) |
| 1 | 0.70802 | 71 | 68 | 70 | | 16 | 0.69459 | 74 | 70 | 68 |
| 2 | 0.6689 | 74 | 68 | 57 | | 17 | 0.73494 | 70 | 67 | 62 |
| 3 | 0.69634 | 71 | 65 | 67 | | 18 | 0.71139 | 72 | 80 | 72 |
| 4 | 0.6636 | 74 | 68 | 60 | | 19 | 0.71611 | 73 | 58 | 70 |
| 5 | 0.7176 | 71 | 75 | 70 | | 20 | 0.73865 | 71 | 67 | 62 |
| 6 | 0.66373 | 74 | 73 | 63 | | 21 | 0.65754 | 74 | 65 | 65 |
| 7 | 0.65147 | 75 | 68 | 58 | | 22 | 0.6923 | 72 | 65 | 62 |
| 8 | 0.72867 | 71 | 67 | 65 | | 23 | 0.70989 | 72 | 75 | 77 |
| 9 | 0.6695 | 74 | 63 | 65 | | 24 | 0.66171 | 75 | 57 | 62 |
| 10 | 0.77997 | 69 | 70 | 72 | | 25 | 0.69638 | 72 | 72 | 77 |
| 11 | 0.74901 | 69 | 73 | 72 | | 26 | 0.71808 | 74 | 62 | 72 |
| 12 | 0.6983 | 74 | 67 | 65 | | 27 | 0.7164 | 71 | 70 | 67 |
| 13 | 0.70219 | 72 | 63 | 72 | | 28 | 0.68226 | 73 | 65 | 68 |
| 14 | 0.69875 | 72 | 68 | 68 | | 29 | 0.71521 | 72 | 65 | 68 |
| 15 | 0.7079 | 73 | 67 | 72 | | | | | | |
| | | | | | | | | | | |
| **MEAN** | loss | train (%) | val (%) | test (%) | | | | | | |
| | 0.701703 | 72.37931 | 67.62069 | 67.17241 | | | | | | |

However, the result is still not good enough even if we expanded our dataset. Our project aims to detect possible heart disease, so we should assume 'murmur', 'extrahls' and 'extrastole' are all the possible indications of heart disease. As a result, we process a more reasonable encoding for our training data classes. The class of 'murmur', 'extrahls' and 'extrastole' were mapped to the same label '1', the class of artifact was mapped to the label '0', and the class of 'normal' was mapped to the label '2'. Hence, the new class label '1' indicates possible heart disease because the label '1' corresponds to the original labels of 'murmur', 'extrahls' and 'extrastole'.

In this way, all our classes are mapped into 3 new categories: '0', '1' and '2'. It will reduce the difficulty in training because the class number was reduced from 5 to 3. Then we train the data in the same model and architecture. As expected, it did have better results.

The loss we got in this way is only 0.57714, which is much lower than the 0.701 before.

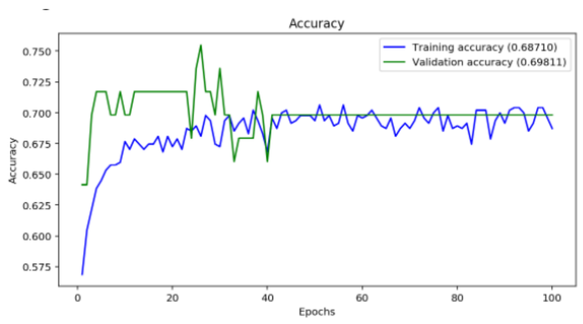
```
Epoch 00092: loss improved from 0.57835 to 0.57714, saving model to ./best_model_trained.hdf5

Epoch 00093: loss did not improve from 0.57714

Epoch 00094: loss did not improve from 0.57714

Epoch 00095: loss did not improve from 0.57714

Epoch 00096: loss did not improve from 0.57714

Epoch 00097: loss did not improve from 0.57714

Epoch 00098: loss did not improve from 0.57714

Epoch 00099: loss did not improve from 0.57714

Epoch 00100: loss did not improve from 0.57714
training finised!
CPU times: user 3min 9s, sys: 22.5 s, total: 3min 31s
Wall time: 1min 47s
```

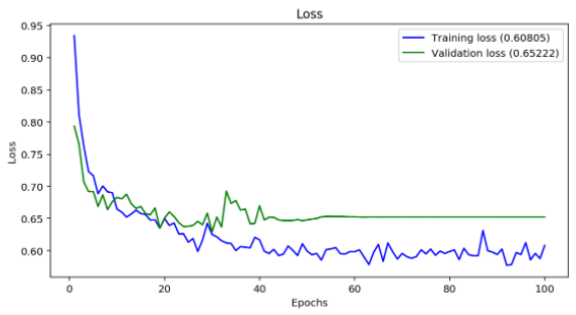
The new training accuracy on the 59 testing cases is  ranging from 77% to 80%, which is also much higher than 67.2% before.

```python
# make a prediction
y_pred = model.predict_classes(x_test, batch_size=32)
#check scores
scores = model.evaluate(x_test, y_test, verbose=0)
print ("Model evaluation accuracy: ", round(scores[1]*100),"%")
```

```
Model evaluation accuracy:  78.0 %
```

The training accuracy and loss reach the expected result in around 40 epoches.





```
CPU times: user 1.53 s, sys: 486 ms, total: 2.02 s
Wall time: 1.21 s
```

+ Code  + Markdown

## Conclusion

Overall, it seems inconclusive to prove that LSTM is a good model to use to classify various heartbeats. One key factor would be the broad classification, each class label (extrahls, murmur and normal) includes many trained audio files that are not equally the same. Hence, minor changes in the heartbeats could have caused the LSTM to incorrectly classify it. Moreover, our training dataset is relatively small, causing the model to not have enough data to work with. Expanding the data set size from 461 to 585 results in about a 2% accuracy improvement,  which indicates that a relative bing data size is necessary. For future work, more datasets and possibly more classification labels could be used. We could also experiment with other models such as RNN and Multilayer Perceptron (MLP).