APPLICATION OF CRNN FOR WHALE A-CALL CLASSIFICATION

A Project Report submitted for DataFest@Integration2023, ISI, *by*

Duo Team- **Humpbackers**

Anannyo Dey (Jadavpur University, Kolkata)

Debasmit Roy (Jadavpur University, Kolkata)

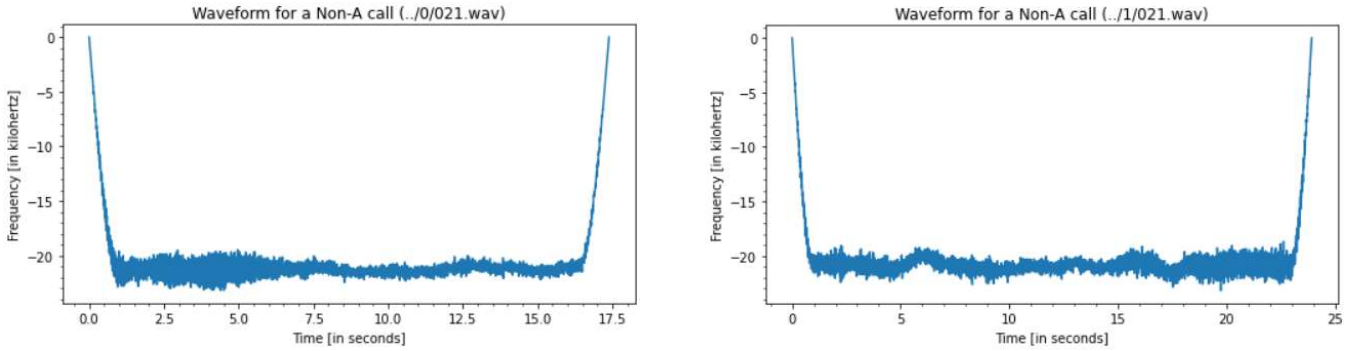
PROBLEM STATEMENT: Binary classification of whale sounds into A-calls and non-A calls by building models that can accurately identify various calls made by these creatures.

DATASET:

Description- The dataset provided was annotated by the DataFest Team at Indian Statistical Institute Kolkata. The occurrences of A calls were identified and labelled from underwater audio recordings captured by deep-sea hydrophone devices. Each candidate is a '.wav' sound clip of 1-channel with a sample rate of 2 kHz. The concise statistics of the durations of the audio clips (in seconds) provided are-

Type of data	No. of samples	mean	std	min	25%	50%	75%	max
Training (Non-A calls)	12951	15.3100	6.5367	6.8875	10.4715	14.9520	18.1520	89.8320
Training (A-calls)	12995	25.3083	6.5279	16.8875	20.4715	24.9520	28.1520	99.8320
Test Data	2000	20.2821	8.1673	7.0155	15.3036	19.1197	25.7195	67.3200

Visualization-



(a) Non A-call waveform (b) A-call waveform

Figure 1: Frequency(kHz)-Time(sec) graphs plotted using Matplotlib package

CHALLENGES:

- The very first major setback we faced was having near-to-none domain knowledge in the field of bio-acoustics that could've helped us in formulating our DL model precisely. Nor did we have any prior experience handling audio files in python, signal processing, creating spectrograms, etc.

- Secondly, we found that the audio files in the dataset provided to us, consisted of a lot of background noise that could be attributed to detectable flow noise from whale movement, nearby ships, other whale calls, etc. This posed a major challenge in the data pre-processing stage to remove noise from the data successfully without losing out on important information.
- We were torn between using CNN models and RNN models for training, as both have their own advantages. CNN models can capture the A-call patterns created in the spectrograms, whereas RNN models can capture the sequential whale-call pulses, frequency sweeps using memorability.

EXPERIMENTS:

Pre-processing and Data Augmentation:

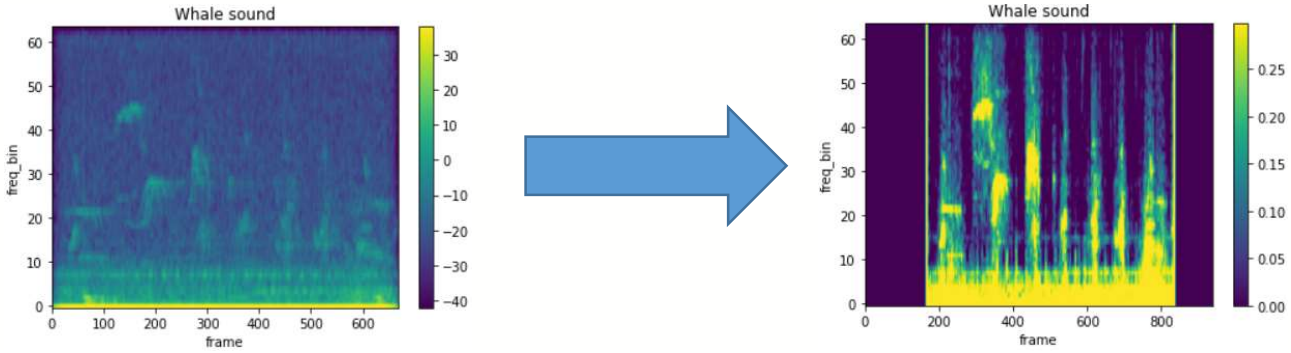
a. We scaled all audio clips into a 30 sec window by truncating longer clips and padding shorter clips.

b. Normalized those signals and applied *Symlet-8 wavelet de-noising* with soft thresholding.

c. Created the log Mel Spectrograms, with *frame length* of 512, *frame steps* 64, and *fft length* of 1024. Using key information provided on the 'Overview' section, about the frequency ranges (70-90Hz) of Blue whale A-calls, we experimented with limiting the range of signal frequencies mapped into Mel scale, and found extracting only 20-150Hz into 64 *mel bins*, yields the best results.

d. We normalized the spectrogram again and applied further de-noising in hope of improvements.

We applied this procedure on all the audio clips provided, both test and train, and stored their spectrograms in '.npy' files for future retrieval.



(a) Raw spectrogram (b) Pre-processed spectrogram

Figure 2: Change in spectrogram of the waveform after pre-processing

Model architecture and Hyper-parameter setting:

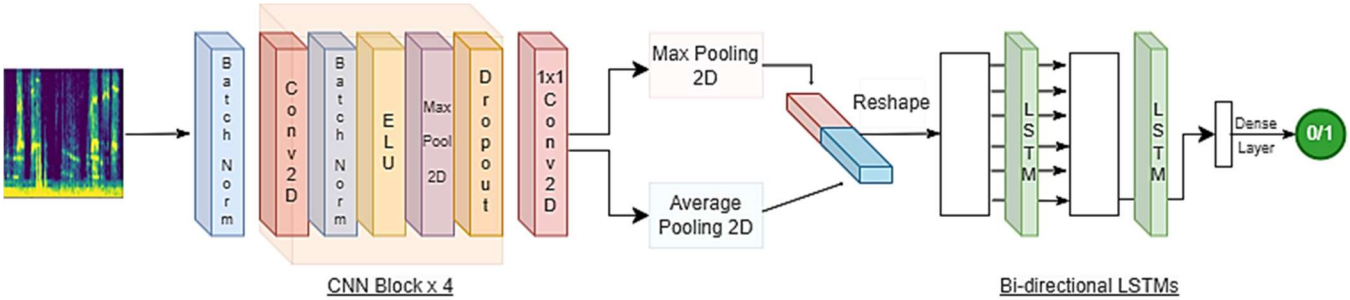
I. CNN

We initially started building CNN models inspired by the existing literature in the field of bio-acoustics. We had trained a 4-block CNN model for 200 epochs in PyTorch with train-validation random split of 80:20 in batches of 16, using Adam optimizer and BCE loss. But irrespective of model tunings, it couldn't exceed 92% accuracy on validation. Thus, we turned to CRNN architecture.

II. CRNN

We created a generator for the spectrograms stored previously, with batch size of 64 and a training-validation random split of 80:20. We constructed a 6layer CRNN model (4 CNN blocks and 2 Bidirectional LSTM layers) in TensorFlow. Each CNN block used Batch Normalization, ELU, MaxPool (2,2), Dropout (= 0.2) layers, and a (True, False) pair for the *return_sequences* of the 2 LSTM layers to get a single output before passing it through a Dense layer using Sigmoid activation. We trained the model for a maximum of 20 epochs using RMSprop optimizer and a BCE loss function, with a custom learning rate scheduler starting from 0.001. We also ran the same using SGD optimizer for 15 epochs or till 98% accuracy was achieved.

Note: We didn't use any pre-trained models for our approach.



RESULTS:

We achieved the following stable scores on the training and validation sets using our model-

Mode	Accuracy	F1-Score
Training	99.96	99.92
Validation	98.63	98.20

After evaluating the same model on the test data set we achieved a micro F1-Score of '**0.98125**' on the Leaderboard (according to the 40% of data).

CONCLUSION:

In this work, we implemented a **CRNN model** using TensorFlow in Google Colab to solve a Binary Classification task of identifying A-calls from Non-A calls of blue whales, with wavelet denoising for increasing the SNR (Signal to Noise Ratio), and minor enhancements for hyper-parameter optimization. We had also explored using CNNs with the same input, but found CRNNs to outperform them in every way. Our proposed model thus performed exceedingly well on both the training and validation sets. It is very much generalizable, as it also yielded competitive F1-scores on the test data as well.

ACKNOWLEDGEMENT:

We wish to take this opportunity to thank the organizers of the DataFest'23 at ISI, Kolkata for hosting these Kaggle competitions, and the Professors and Industry experts for their valuable time, who would be kindly evaluating our work. Thanks for providing us with such a unique challenge to learn, brainstorm, experiment, and gather experience in totally unexplored domains.