Section: Research Paper
ISSN 2063-5346

Automated Capuchinbird Call Detection with CNN Based Audio Classifier

Poonam Kadian[1] , Anshul Pareek [2], Shaifali M.Arora[3]

[1] Maharaja Surajmal Institue of Technology, New Delhi,India.: poonam.dahiya@msit.in
[2] Maharaja Surajmal Institue of Technology, New Delhi,India.: er.anshulpareek@msit.in
[3] Maharaja Surajmal Institue of Technology, New Delhi,India: shaifali04@msit.in (Corresponding Author)

***Abstract*— Bird density prediction plays an important role in monitoring and further protecting biodiversity. Recent advances in acoustic sensor networks and deep learning techniques provide a novel way for continuously monitoring birds. In this project we aim at devising a method to measure the density of capuchin birds, natively found in rainforests of Amazon. We will be using forest audio clips to count the capuchin bird calls, thereby predicting the Capuchin bird density inthe forest. To count the Capuchin bird calls, we will be using a CNN (Convolutional Neural Network) model which feeds on audio spectrogram to recognize Capuchin bird calls. The major libraries required to build and train our model are tensorflow, numpy array and keras.**

Keywords*— *capuchin classifier, binary classification, 2D CNN, short time fourier transform, logarithmic amplitude scale spectrogram, data augmentation.

1. INTRODUCTION

Birds help with the pollination of flowers, elimination of pests, scavenging dead animals, thereby helping in the rejuvenation of forests. It is essential to understand the relevance of birds to ecosystems in order to appreciate the services they offer [1]. Besides, in comparison to other vertebrates, birds serve as an extremely reliable indicator to determine how susceptible they are to environmental toxins. Therefore, birds diversity is an indicator of species richness of natural ecosystem [2]. Thus regular monitoring of species population in a certain forest area can be a crucial step towards determining the forest's health and for developing steps for conservation species with highest priority [3].

Capuchinbirds (*Perissocephalus tricolor*) are natively found in the rainforests of North-eastern South America, almost north of Amazon River. They have a fairly large sized bird having a pale bluish grey face and cinnamon brown body, which accounts for easier task of spotting capuchinbirds. Though Capuchinbirds are quite solitary, but when found in lek, usually around 6 to 8 males gather in the sub canopy regions. They mainly consume upon fruits but also rely upon large insects, ensuring a rich species diversity hanging around them. Male capuchinbirds in lek, create synchronised moo-sounds during lek, adjusting the timing of their calls such that they follow other males of the lek without overlapping. Due to features like unmistakable body appearance, most commonly found in sub canopy and middle levels of forest, having a and a distinct moo-sound for communication makes it easier for ecologists to monitor these birds and study/evaluate the ecological diversity around their habitats.

Because of the capuchinbird's large and stable population in the rainforests of South America, ecological studies and surveys monitoring them have been minimal to non-existent in the past few decades. As a result, rarely any dedicated automatic classifier exists to parse Capuchinbirds. In this paper,we describe our approach to training a deep learning convolutional network classifier that particularly focuses on binary audio classification of Capuchinbird calls. Even though rarely any surveys data has been conducted to monitor capuchinbirds and if done they don't provide open access license to use the recordings, still we managed to gather a small dataset of Capuchinbird recordings on Kaggle provided in the form of 'Z by HP Unlocked Challenge 3 - Signal Processing' challenge. Due of our limited knowledge, we have no notion which audio clip characteristics to employ for categorization purposes;

here is where a convolutional neural network (CNN) model comes in handy for classification purpose. Through this paper we aim at developing several models distinguishable upon following two factors, (i) inclusion/exclusion of data augmentation and (ii) amplitude scaling techniques. And later compare their performance on several reliable metrics, to bring forward a general purpose automated audio classifier for capuchinbirds.

2. REALTED WORK

Deep learning models have been shown to offer a significant amount of potential in the field of classification issues, such as object detection and picture segmentation, over the course of the past ten to fifteen years. This potential has been demonstrated through a number of different experiments. In addition, there has been a consistent increase in interest in employing acoustics for the purpose of ecological surveying over the course of the past decade. This increase in interest has been made possible by a decline in the prices of recording hardware such as ARUs and software to process the data collected,as well as by advancements in the automated species density analysis [4].

Before deep learning gained wide-spread popularity, prior work had focused on feature extraction from raw audio recordings, followed by some classification models, such as Random Forest [5] and Support Vector Machines [6]. A major issue with these models was that in order for the learning algorithms to be effective, most of the features have to be identified manually by a domain expert. In contrast, Deep Learning methods function acts like a black box: you feed them the input in the form of spectrograms, and the convolutional layers automatically do the feature extraction for you. Deep learning based neural networks frequently outperform traditional techniques in the domain of bioacoustics event detection and classification.

With the increasing volume of available audio recordings and the development of machine learning algorithms, autonomous classification of animal sounds has recently attracted a wide range of interests. Such models are being used to passively monitor Hainan gibbon calls, one of the world's rarest primate [7] and Northern Bobwhite calls [8], to monitor the presence of whale acoustics [9], as well as giant deep learning models such as BirdNET which is capable to classify up to 984 bird species [10]. For a number of years, deep learning techniques have dominated these automated analysis approaches. The results of the yearly LifeCLEF is a good testimonial for the above statement. It's a Bird Detection Challenge held annually, which was first held in 2014. There are as many as 1500 different species of birds, making this one of the most stressful bird sound identification challenges to solve. In the further years of the LifeCLEF competition, it was also shown that shallow neural networks with only a few layers [11] couldperform on par with very deep architectures [12].

In order to extract complex characteristics from spectrograms and distinguish possible comparable classes, certain well-known pre-trained CNN architecture based deep models, such as VGG1624 and ResNet25, are available. Due to the fact that these models require tens of millions of parameters for training, we decided against employing them because our dataset is quite tiny. Since we only have 810 bird-related audio files in our dataset, we choose to use shallow architecture based models to train for capuchin audio classification. The models proposed on this paper feed upon input in the form of spectrograms extracted from 1 second audio clips.

3. METHODOLOGY

Our overall workflow for training a deep neural network classifier consisted of the acquisition of Substantial amount of audio data clips, depending upon the model type applying data augmentation or skip this phase, followed by data pre-processing of this data to generate visual representations of sound suing linear or logarithmic amplitude scaled spectrograms, and, finally, the training of a complex model architecture with ~1.2 million trainable parameters. We ran inference on independent validation and test splits. The test data also consisted of a separate class of augmented audio clips using overlay technique

to mix parsed audio clips with non_parsed capuchin clips. These clips weren't used for training purpose. The workflow can be further subdivided into following categories-

3.1 Data Collection

The dataset can be found on Kaggle website as part of the 'Z by HP Unlocked Challenge 3 - Signal Processing' challenge. This dataset comprises of three directories: (i) Parsed Capuchinbird Clips, (ii) Parsed Not Capuchinbird Clips and (iii) Forest Recordings. To train the models we use first two sub directories. Each audio clip has a sampling rate of 44100Hz and consists 2 channels, one of these channels has to be dropped during audio-processing stage.

The Parsed Capuchinbird clips' directory consists of 217 audio clips, with audio length ranging from two to five seconds with an mean audio clip length of 3 seconds. On the other hand, Parsed Not Capuchinbird clips' directory consists a total of 593 files. The Imbalance ratio between these two classes is 2.73 which clearly indicates presence of class imbalance.

3.2 Data Augmentation

We wanted to see how much a naive CNN model's performance gets affected by presence of moderate to heavy class imbalance. Thus for two models (LiSP and LoSP) we preferred to skip this phase of applying data augmentation. For the other two models (LiSP_AUG and LoSP_AUG), we used the SMOTE technique of data augmentation to increase the number of audio samples of 'parsed_clip' class. There are two data augmentation techniques in a CNN audio classification: (i) Raw Audio Augmentation,and (ii) Spectrogram Augmentation. In our models we planned to stick with the former method i.e. usingvarious Raw Audio Augmentation techniques. The 3 raw audio augmentation techniques used in our architecture are: Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

3.3 Noise injection

The goal of this technique is to inject random values into the audio data. To achieve this we used 'randn' object present inside 'numpy.random library'. To still hear the audio clip clearly, we limited the amplitude of the noise.

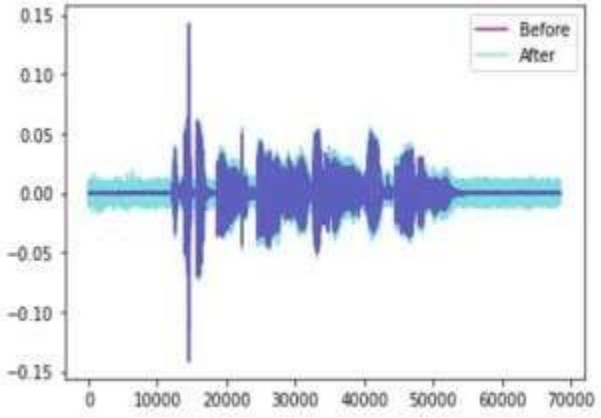

Fig. 1. Noise injection

3.4 Pitch shifting

It is how pitch scaling is implemented in musical instruments. It is a method of altering sound's pitch without altering its speed. We're going to use the pitch shift function from librosa once more. It requires wave samples, sample speed, and the quantity of pitch-shifting steps. Depending on the predetermined pitch interval, the signal's pitch is either raised or lowered, without affecting the time axis. In simpler terms, it aims at modifying the frequency parts of the audio. We altered the pitch by a factor lying within range of $\{1, 4\}$ to ensure that audio clip still fits in the 'parsed_clip' class. To introduce pitch shift in audio clips, we used the 'pitch_shift' object present in librosa.effects library.

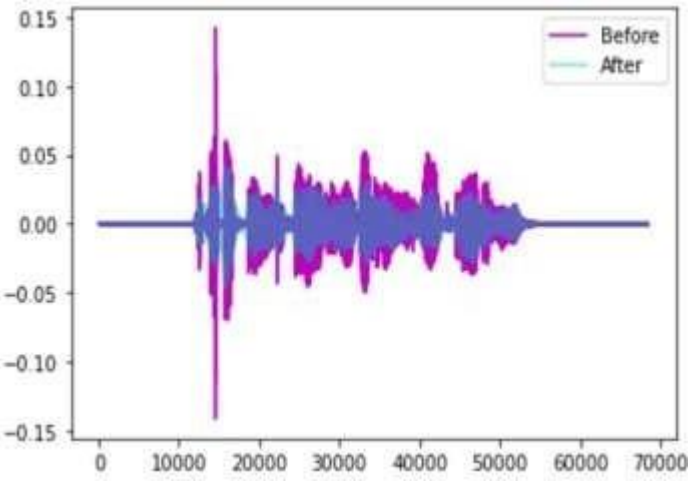

Fig. 2. Pitch injection

3.5 Time shifting

The It is the method of altering sound's speed/duration without modifying its pitch. The time stretch function in librosa can be used to alter the speed of any audio clip. Time_stretch function takes wave samples and a factor by which to stretch as inputs. A delay of up to a factor of 0.8 was introduced into the original clips, and on the contrary came handy in reducing the audio clip length by introducing a factor of 1.25. For introducing time stretch in audio clips we used 'time_stretch' object present inside 'librosa.effects'.

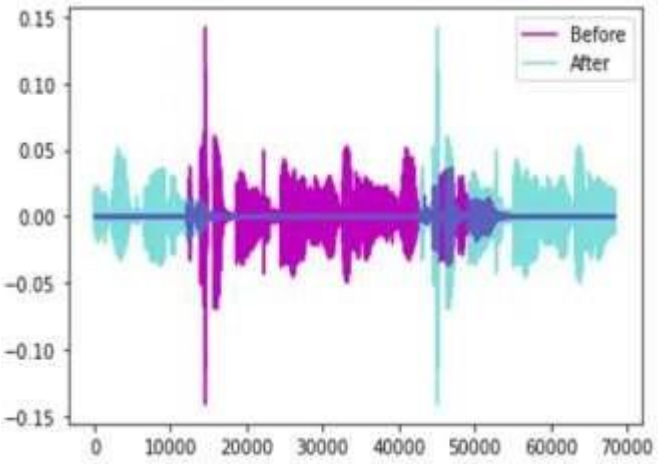

Fig. 3. Time Shifting

During our experimental analysis, we observed pitch shifting to be the best augmentation technique as by scaling down the pitch through several semitones we can produce calls that sound similar to calf calls, that ensures a robust training dataset for the proposed CNN architecture models (LiSP_AUG and LoSP_AUG).

3.6 Data Pre-Processing

We started this phase by reading the audio files in ".wav" format, into a numpy arrays. Followed by converting them from stereo to mono channel audio clips. It was ideal dropping one channel (by taking the mean of the two channels) in order to reduce unnecessary information that increases training data size unnecessarily. After that, we standardize the sampling rate of audio clips, otherwise two audio clips having same length but different sampling rate will produce numpy arrays of varying length. Thus, for standardization we brought down sampling rate from 44100Hz to 22050Hz. Besides, even though

44.1kHz offers higher audio quality, 22.05kHz is sufficient for audio classification. Additionally, if the amount of each piece of data is reduced to half, the model can be trained more quickly. Once, resampling is done, we resize each audio clip to same length i.e., 1 second, by either truncating audio clips having length greater than 1 seconds or padding by padding it with zeroes if length was shorter than 1 second. Once, we had standardized our audio data stored in numpy arrays, then we extract the spectrograms out of these numpy float64 arrays. A spectrogram can be described as a picture formed by joining together, multiple wave spectrums taken at varying instances of time.

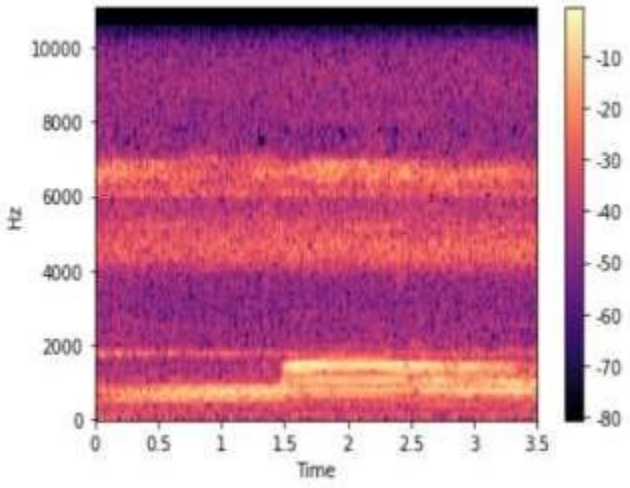

Fig. 4. Spectrogram of Capuchinbird call

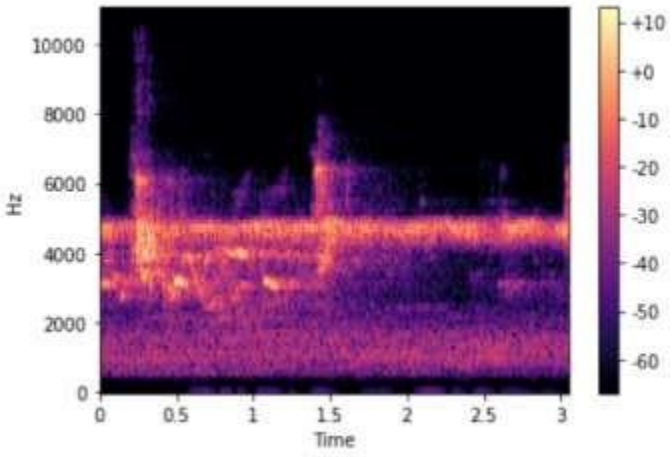

Fig. 5. Spectrogram of random bird call

In the above picture, Fig. 4. displays audio signal in amplitude vs time domain whereas Fig. 5. displays same audio signal in frequency vs time domain. It offers us a feel of how loud or silent a clip is at any one time, but it doesn't reveal us a lot about the frequencies that are present. In previous works it has been shown that the frequency-time domain signal representation using Short Time Fourier Transform (STFT) performs significantly better than amplitude-time domain signal. With the help of librosa library, from processed audio waveforms we extracted the spectrograms into numpy arrays of complex datatype. The two crucial parameters for librosa.display.specshow() method, i.e. frame-length and hop-length were chosen as 256 and 129 respectively. Converted the array into absolute values. Depending upon the model architecture, we further converted amplitude scale into dB scale (for LoSP/LoSP_AUG models) or left the amplitude scale unchanged (for LiSP/LiSP_AUG models) and added a dimension to portray them as greyscale images.

Section: Research Paper
ISSN 2063-5346

On completion of this phase, we obtain float numpy arrays of shape (129, 171, 1) for each audio spectrogram stored in python list. These lists are converted into tensorflow datasets and then batched along with binary labels: 1 to label parsed capuchin call and 0 to label non_parsed capuchin calls.

3.7 Model Training

For all the four models: LiSP, LiSP_AUG, LoSP and LoSP_AUG, the underlying CNN architecture remains the same. The proposed CNN architecture in this paper can be subdivided into two parts, Feature Identification and Classification.

3.8 Feature extraction part

The former sub-architecture is built by connecting two stacked layers. First sub-layer consists of a 2D convolutional layer along with a 2D average pooling layer of size 2x2. The convoutional layer consists of 64 kernels having a filter size of 4x4 each. Average pooling layer takes average value of a block in a feature map, thereby resulting in a downsampled feature map. Since former stacked layer directly feeds upon input data. During pooling we prefer each downsampled element obtained throuh pooling, to have an imact from all neighboring values of its region. Thus, Average pooling correctly serves the purpose for former stacked layer. The latter stacked layer consists of another 2D convolutional layer having only 32 kernels each of size 4x4. Interestingly, we choose a Max pooling layer of size 3x3 here at this layer we want to extract the most important features only.

The kernels (weights) help extract characteristics which identify different images from one another, exactly what we want for classification. There are links within the architecture between the output of previous layers and convolutional layers. Each link represents a distinct kernel, and the convolutional neuron computes an elementwise dot product of this distinct kernel with the output of the preceding layer's correspoding neuron. The dot product operation employs a stride of one, simply implying that thekernel is shifted by one pixel per dot product. After each convolutional layer, we apply a non-linear activation function since, a convolutional network is linear in nature but the problem we are dealing withis nonlinear in nature. We chose the ReLU function (Rectified Linear Unit) as activation function for our the concolutional layers.

$$ReLu\ (x) = \ max\ (0, x)$$

ReLu function being a one-to-one mathematical operation, it is a simple mathematical expression, thus helps in faster learning of model and improving results on the gradient. After activation function, we pass them through the Average and Max pooling layers depending upon their respective stacked sub-layer.

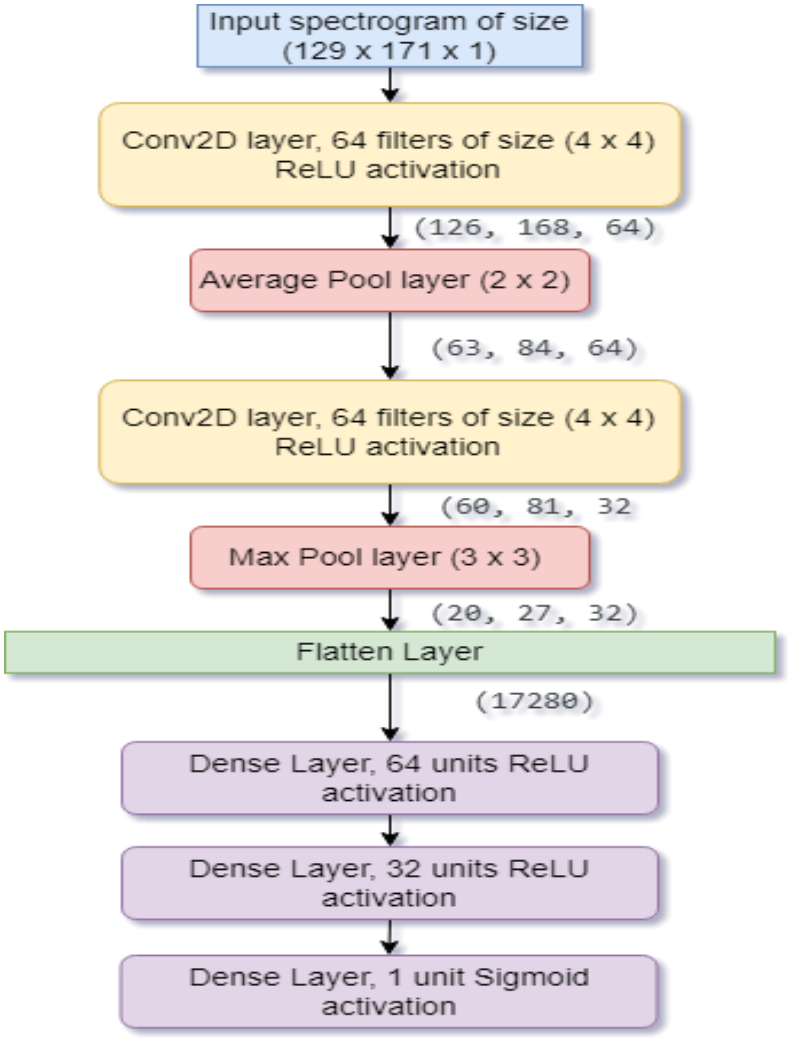

Fig. 6. CNN architecture for deep learning models

3.9 Classifiaction part

Before feeding the final output of the above network architecture, we have to pass it through the process of flattening. Flattening is an intermediate process which is present in between the final pooling layer of the convolutional network and the first dense layer of the connected neural network. The sole purpose of this layer is to turn a three-dimensional network layer into a one-dimensional vector in order to suit the input specification of a fully-connected layer, which will be later used for classification purpose. This linear vector obtained from flatten layer is passed as input to the Fully Connected Neural Network. The fully connected network of neurons consists of three dense layers of neurons. First dense layers consist of 64 neurons, followed by another dense layer of 32 neurons, both layers having a ReLu activation function. The final layer consists only a single neuron having sigmoid activation function, acting as the output layer of the proposed CNN architecture. Sigmoid activation function is the ideal choice for the output layer, since we are dealing with a binary classification problem.

To train the proposed models in this paper using the above defined CNN architecture, we have used Keras Sequential API since it is easier to implement, where you keep stacking one layer after another. Sequential API is suitable for such shallow architectures dealing with three to 5 layers. The model is compiled using ‘binary_crossentropy’ as the loss function for binary classification problem along with

'Adam' optimizer. The number of parameters remains unchanged ~1.2 million, regardless of inclusion or exclusion augmented data within the architecture.

4 RESULTS

In this study, a CNN architecture is proposed, which is further used to train four different models. These models are built using Keras (Deep Learning Library), Tensorflow datasets and Quadra P4000 GPU. Accuracy, Precision, Recall and F1 Score are the 4 metrics used for evaluating the performance of the four derived models LiSP, LiSP_AUG, LoSP and LoSP_AUG. These models even though sharing same inner lying architecture as discussed in the methodology, can be differentiated on the basis of inclusion/exclusion of augmented data and different amplitude scaled spectrograms used to train each CNN model. The result can be summarized in the form of a table shown below:

TABLE I. EVALUATION OF MODELS

Models	**Metrics (in %)**			
	Accuracy	***Precision***	***Recall***	***F1 Score***
LiSP	62.8	68.96	22.72	34.18
LiSP_AUG	80.78	93.06	69.11	79.32
LoSP	86.47	91.66	75.00	82.50
LoSP_AUG	94.56	95.38	94.65	95.01

From Table I, we can clearly see, the worst performing model is LiSP in which data augmentation stage was dropped alongside using a linear scale amplitude spectrogram. On the contrary, the LoSP_AUGmodel using data augmentation as well as, a log scale amplitude spectrogram performs the best among all4 mentioned models in this paper. Below are the plot of Loss value vs epoch during training for every model:

This paper also throws light on the fact that inclusion of a log scale spectrogram induces improved accuracy, whereas the inclusion of data augmentation techniques helps improve F-beta score of a model. Besides, on comparing the Loss function graphs, it can be noticed that figure 4.4, Loss of LoSP_AUG model, depicts the smoothest learning curve among all the 4 models proposed in this paper and LiSP_AUG has the worst learning curve. Often times, the loss vs epoch plot reveals a lot about the learning phase of the model like how smooth was the leaning and whether a model under training goes into overfitting or not. Below are the plots of loss vs training epochs for the four models:

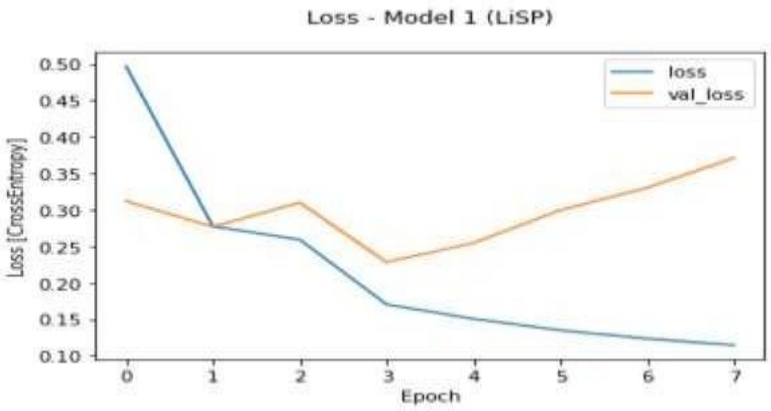

Fig. 7. Loss v/s epoch plot of LiSP model

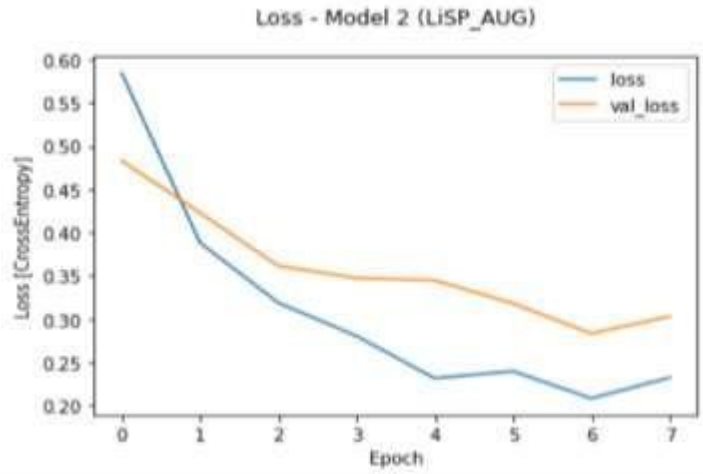

Fig. 8. Loss v/s epoch plot of LiSP_AUG model

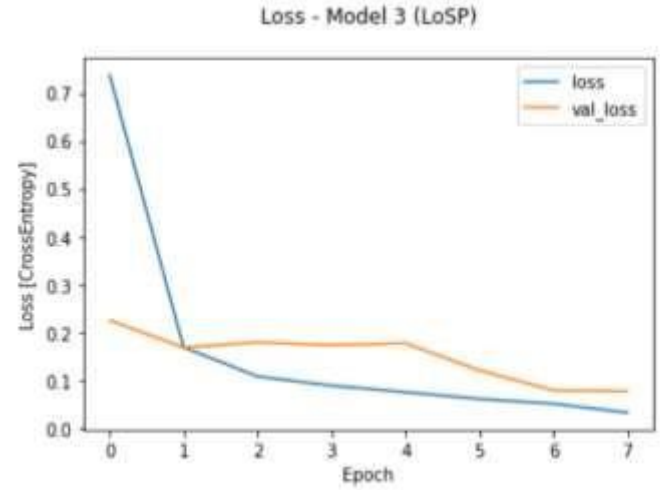

Fig. 9. Loss v/s epoch plot of LoSP model

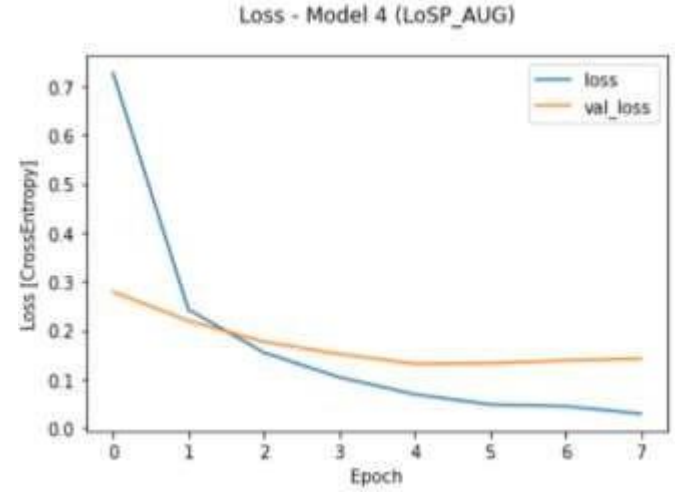

Fig. 10. Loss v/s epoch plot of LoSP_AUG model

From the Fig. 7, Fig. 8, Fig. 9 and Fig. 10, we can observe that the learning curves of LiSP_AUG and LoSP_AUG models are more smooth in comaprison to the other two models. We can also observe

the fact that first two models based upon linear amplitude scale spectrogram have overfitted upon training data.

5 CONCLUSION

In this paper we proposed an architecture based upon Convolutional Neural Network, by converting audio signals into spectrograms after performing STFT, for capuchin-bird audio classification. On the top of this CNN architecture, we built four different models: LiSP, LiSP_AUG, LoSP and LoSP_AUG. Through these models we found that log-scale amplitude spectrograms performed significantly better in comparison to linear-scale amplitude spectrograms. We also witnessed improved performance with the introduction of augmented data. The combination of log-scale amplitude spectrograms and augmented data outperformed the other three CNN models. Finally, we individually examined the influence of augmentation and log-scale spectrograms on the CNN model with respect to a simple linear spectrogram-based model. We observed that log-scale spectrograms significantly improved the model's accuracy whereas, data augmentation helped in improving the lower F1 score and recall metrics of the model due to presence of moderate dataset imbalance.

REFERENCES

[1] C. Sekercioglu and D. Wenny 2016 Why Birds Matter. *Avian Ecological Functions and Ecosystem Services*.
[2] Egwumah, F. A., P. O. Egwumah, and D. I. Edet. 2017 Paramount roles of wild birds as bioindicators of contamination. *Int J Avian & Wildlife Biol* 2.6: 00041.
[3] S. Kahl, Connor M. Wood, M. Eibl and H. Klinck 2021 BirdNET: A deep learning solution for avian diversity monitoring *Ecological Informatics*. 61:101236.
[4] J. Xie, K. Hu, M. Zhu, J. Yu and Q. Zhu 2019 Investigation of Different CNN-Based Models for Improved Bird Sound Classification *IEEE Access*. 7:175353-175361.
[5] D. Stowell and M. D. Plumbley 2014 Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning *PeerJ*. 2: e488.
[6] R.O. Tachibana, N. Oosugi and K. Okanoya 2014 Semi-Automatic Classification of Birdsong Elements Using a Linear Support Vector Machine *PLOS ONE*. 9(3) :e92584.
[7] Dufourq, E., Durbach, I., Hansford, J.P., Hoepfner, A., Ma, H., Bryant, J.V., Stender, C.S., Li, W., Liu, Z., Chen, Q. and Zhou Z. 2021 Automated detection of Hainan gibbon calls for passive acoustic monitoring *Remote Sensing in Ecology and Conservation*. 7:475-487.
[8 Nolan, V., Scott, C., Yeiser, J. M., Wilhite, N., Howell, P. E., Ingram, D. and Martin J. A. 2022 The development of a convolutional neural network for the automatic detection of Northern Bobwhite Colinus virginianus covey calls *Remote Sensing in Ecology and Conservation*.
[9] C. Bergler, M. Schmitt, A. Maier, R. X. Cheng, V. Barth and E. Nöth 2022 ORCA-PARTY: An Automatic Killer Whale Sound Type Separation Toolkit Using Deep Learning ICASSP 2022 - 2022 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore*: 1046-1050.
[10] J. Schlüter 2018 Bird Identification from Timestamped, Geotagged Audio Recordings *CLEF (Working Notes)*, 2125.
[11] M. Lasseck 2018 Audio-based Bird Species Identification with Deep Convolutional Neural Networks *CLEF (working notes)*, 2125.

Poonam is an Assistant Professor at MSIT, New Delhi. She received her B.Tech degree in Electronics and Communication Engineering from Kurukshetra University, India, M.Tech degree in Digital Communication from Shobhit University, and Ph.D. from G.D. Goenka University, Gurgaon, Haryana. She has an experience of more than 13 years and her fields of interests are: Wireless Communication, IoT, Image Processing, Machine learning, Artificial Intelligence.
E-mail: poonam.dahiya@msit.in

Anshul Pareek is a computer vision researcher interested in finding applications of computer vision in different areas such as mobile robotics, nuclear and medical field. **She** received her B.E. in ECE from University of Rajasthan, India in 2004. She completed her M.Tech. in Digital Communication from Rajasthan Technical University in 2012 and Ph.D. in Computer vision and mobile robotics in 2022. Presentlyworking as Assistant Professor in Maharaja Surajmal Institute of Technology, New Delhi, India
E-mail: er.anshulpareek@msit.in

Section: Research Paper
ISSN 2063-5346

Shaifali M. Arora is currently working as Associate Professor at Maharaja Surajmal Institute of Technology, New Delhi. She has completed her Ph.D from GGSIPU, New Delhi in 2017. She has a teaching experience of more than 20 years. She has published more than 50 research papers in various reputed journals and conferences. Her areas of interest are Electronics & Communications Engg., Signals, Image and Video Processing, Embedded Systems, Artificial Intelligence and Machine Learning.
E-mail: shaifali04@msit.in