Realtime bird sound detection using recurrent neural networks

Saarthak Verma*[1] and Dr Deepak Kumar Singh[1]

[1]Department of Computer Science, Indian Institute of Information Techology Lucknow
[2]Name of the group, Name of the industry

Abstract—**Bird sound detection and classification is crucial for ecological monitoring and enhancing bird-watching experiences, but traditional approaches face challenges like background noise and overlapping vocalizations. This research develops a real-time bird sound detection system using deep learning, specifically a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) architecture trained on mel-spectrograms of bird sounds. The proposed RNN approach outperforms Convolutional Neural Networks, achieving 75% accuracy by effectively capturing temporal dependencies and learning discriminative spectro-temporal features of bird vocalizations. The system can be deployed for applications like ecological studies, conservation efforts, and enhancing bird-watching through real-time detection and classification, contributing to the field of bioacoustics and environmental monitoring.**

I. Introduction

THE target audience comprises nature enthusiasts and ornithologists seeking a practical means to differentiate bird species solely through auditory traits. This is particularly valuable given the difficulty of spotting elusive songbirds in their natural habitat. Our research on real-time bird sound detection employs this methodology to facilitate bird identification in forests and bird-watching spots. The system captures live audio through a microphone, processes the signal to reduce noise, and converts the cleaned audio into a spectrogram. This spectrogram is then fed into a recurrent neural network (RNN) trained to recognize bird species based on their vocalizations.

In dense forests, visual identification of birds is often hindered by foliage and the birds' elusive nature. By deploying our real-time bird sound detection system, we can continuously monitor and identify bird species based on their calls and songs. The system can be mounted on stationary units or integrated into portable devices for researchers and enthusiasts. As birds vocalize, their sounds are recorded, processed, and analyzed, providing immediate feedback on the species present. This capability is invaluable for ecological studies, biodiversity assessments, and conservation efforts, allowing for non-invasive monitoring of avian populations.

For bird watchers, identifying birds by sound adds a new dimension to the experience, especially in popular bird-watching spots where multiple species may be present simultaneously. Our system can be used to enhance guided tours, providing real-time identification of bird species through a mobile app or a handheld device. As bird watchers explore the area, the system captures ambient bird sounds, processes them, and displays the species identification along with additional information about the birds. This not only enriches the bird-watching experience but also serves as an educational tool, fostering a deeper appreciation and understanding of avian diversity.

Although audio signals are inherently one-dimensional, images are two-dimensional, necessitating a suitable transformation for compatibility. For this purpose, we employ spectrograms, which visually represent the magnitude obtained through the Short-Time Fourier Transform (STFT)[1]. The STFT is an adaptation of the Discrete Fourier Transform (DFT) that, instead of performing a single DFT on a lengthy signal, divides the signal into partially overlapping segments and applies the DFT to each segment using a sliding window. This process produces a two-dimensional spectral representation of an audio segment, with time and frequency as the axes. The spectrogram visualizes the STFT output using a color map, allowing it to be treated as an image that can be fed into a pre-trained image-based network.

II. Literature Review

Bird sound classification has long been a challenging task due to the variability in bird songs and the presence of background noise in field recordings. Over the years, various approaches and methodologies have been explored to tackle these challenges.

The task of bird song classification using neural networks dates back to 1997 when McIlraith and Card investigated bird song identification using artificial neural networks and statistical analysis. Their work demonstrated the feasibility of using neural networks for bird song classification, laying the groundwork for future research in this area [2].

In 2011, Neal et al. utilized features such as binned frequency spectrum, Mel-frequency cepstral coefficients (MFCC), and linear prediction coefficients (LPC) for bird song classification. They employed an ensemble of logistic regression, random forests, and extremely randomized trees, achieving fourth place in the NIPS4B bird classification challenge hosted on Kaggle. This study highlighted the effectiveness of combining multiple feature types and classifiers to improve performance in noisy acoustic environments [3].

Leng and Dat, in 2014, proposed a random forest-based segmentation method to select bird calls in noisy environments, achieving a remarkable 93.6% accuracy. Their approach demonstrated the robustness of ensemble methods in handling noisy data [4].

In the same year, Stowell and Plumbley explored the use of unsupervised learning for audio-only bird classification, therefore training large and diverse datasets without the need for extensive labeled data [5].

The BirdCLEF competition is an annual machine learning challenge focused on developing models for bird sound identification. It is part of the larger LifeCLEF lab, which aims to advance research in biodiversity and environmental monitoring through the application of machine learning and artificial intelligence.

The BirdCLEF 2017 dataset, consisting of over 36,000 audio files from 1,500 different species collected from Xeno-canto, presents a significant challenge due to its uneven class distribution and the need to recognize both single audible species and multiple overlayed sounds in field recordings. Kahl et al. conducted experiments on this dataset, achieving 60% accuracy for overlayed sounds and 68% for recognizing the dominant species using convolutional neural networks (CNNs) [6].

In 2018, Lostanlen et al. developed a method utilizing scattering transforms for robust bird song classification in noisy environments. They captured the hierarchical structure of bird songs, leading to improved classification performance [7].

More recently, in 2019, Stowell et al. introduced a novel approach using unsupervised learning techniques combined with domain adaptation to enhance bird sound classification performance. [8].

III. Research Methods

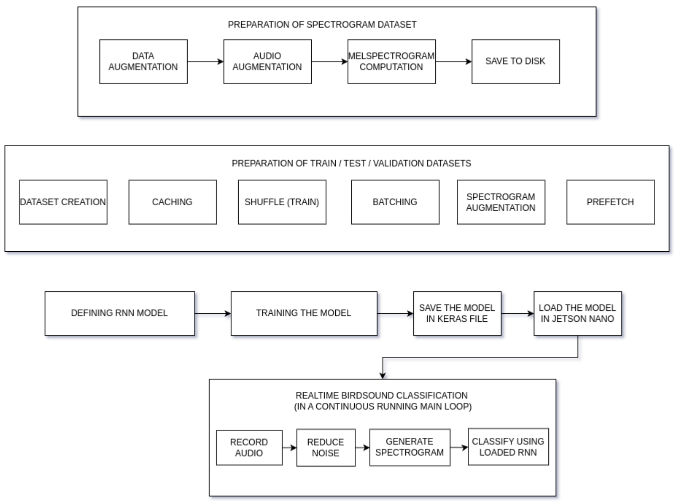

PREPARATION OF SPECTROGRAM DATASET
DATA AUGMENTATION
AUDIO AUGMENTATION
MELSPECTROGRAM COMPUTATION
SAVE TO DISK
PREPARATION OF TRAIN / TEST / VALIDATION DATASETS
DATASET CREATION
CACHING
SHUFFLE (TRAIN)
BATCHING
SPECTROGRAM AUGMENTATION
PREFETCH
DEFINING RNN MODEL
TRAINING THE MODEL
SAVE THE MODEL IN KERAS FILE
LOAD THE MODEL IN JETSON NANO
REALTIME BIRDSOUND CLASSIFICATION (IN A CONTINUOUS RUNNING MAIN LOOP)
RECORD AUDIO
REDUCE NOISE
GENERATE SPECTROGRAM
CLASSIFY USING LOADED RNN

Fig. 1. Methodology used

The methodology used in this research can be broadly classified in the following steps:

- Preparation of spectrogram dataset
- Preparing training testing and validation datasets
- Modeling
- Training of model
- Real time inference of bird sounds

A. *Preparation of spectrogram dataset*

We are using BirdCLEF 2023 competition dataset.

1) **Data Augmentation**: The audios are cropped or padded to make them 10 seconds in length(determined from experimentation)
2) **Audio Augmentation**: We add noise to the input by drawing samples from a random Gaussian distribution. Then we keep both this noisy and not noisy data for furthur steps.
3) **Short-Time Fourier Transform (STFT)**: We apply STFT [1] with a window size of 520 and a Hamming window with an overlap of 260 (half of the window size, configurable. The numbers presented here are determined from experimentation).
4) **Spectrogram Representation**: After the STFT, we retain only the magnitude component, as the phase information is not needed for the spectrogram representation.
5) **Mel-Spectrogram Generation**: We compute mel-spectrograms from the above spectrograms and save them to disk as a JPEG image, creating a mel-spectrogram dataset. Regular spectrograms (STFT) show frequency over time. Mel spectrograms are made by transforming this to the mel scale (perceived pitch) and applying mel filters. These filters emphasize lower frequencies, mimicking human hearing, and their outputs are summed to create the final mel spectrogram. Research has show that melspectrograms work much better than normal spectrograms[9][10][11]. Steps 3 to 5 can be done by using the librosa python library.

The dataset is uploaded on kaggle. The first version of the dataset used TIFF image format, the resulting dataset was 14GB. This version did not include cropping and padding. This version also did not include noise.

The second version of dataset used all the preparation methods mentioned above and stored in spectrogram in JPEG format. The total size of this dataset is 2GB. A major improvement from the previous dataset.

If the spectrogram are saved to the disk by sequentially processing the audio files then it will take hours and hours to do so. Using modern multicore CPUs and asynchronous programming we can reduce this time to 1 hour. To achieve this we use a process pool (usually the number of process spawned is equal to the number of cores, in my experimentation I used a dual core CPU). The dataset is equally distributed between the processes. Each process with its slice of the dataset overlaps the execution the following through asynchronous operations:

- Load the audio file (I/O operation)
- Compute the mel-spectrogram (CPU operation)
- Save the spectrogram to disk (I/O operation)

The overlapping of these ensure that the CPU is always busy and is not bottle-necked by the IO operations. We limit the number of audios that are held in the memory, as the blindly loading the audio files will fill up all the RAM, leading to OOM (Out of Memory) exception, halting execution.

The neural network in the later section will be trained on these spectrograms. This neural network doesn't accept audio input directly. This approach offers several advantages:

- It allows for quick iterative changes to the neural network architecture, as neural networks that take audio as input are bottlenecked by the loading of audio files and the computation of mel-spectrograms (which happens on the CPU in librosa, therefore cannot take advantage of the GPU).
- Having spectrogram images stored on disk enables faster training using the GPU.
- Caching, batching, and prefetching using the TensorFlow dataset library [12] allow for very fast training of the neural network.

B. Preparation training testing and validation datasets

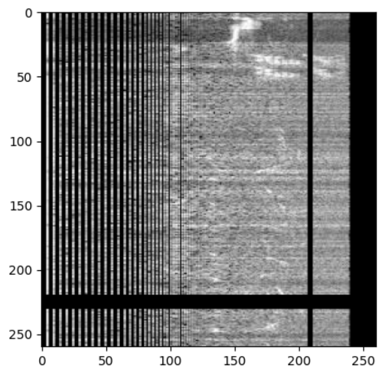
The dataset is prepared for the neural network using TensorFlow's tf.data [12] API. This method ensures efficient data loading and preprocessing through steps such as batching, prefetching, and shuffling (for the training dataset). Various data augmentation techniques, including time and frequency masking, cut mix, and mix up, are also applied to enhance the model's robustness and generalization capabilities.

1) **Dataset creation**: The dataset is initially created from the provided file paths(path of the spectrogram image) and corresponding labels, converting these into a TensorFlow dataset. We upsample the dataset to ensure that the population of less-represented classes crosses a certain threshold.
2) **Caching** : Caching is employed to speed up data loading. By storing the dataset in memory after the first epoch, caching reduces the overhead of repeated I/O operations, thus improving overall training efficiency.
3) **Shuffling (Training Only)** For the training dataset, shuffling is applied to ensure that the model sees a random order of data in each epoch. This randomness helps in mixing the data points effectively, which improves the model's ability to generalize and prevents overfitting to the order of the data.
4) **Batching** Batching divides the dataset into smaller groups of samples. This allows the model to process multiple samples in parallel, increasing the training efficiency and making better use of computational resources.
5) **Spectrogram Augmentation** : Time masking obscures certain portions of the spectrogram along the time axis, while frequency masking does so along the frequency axis. This augmentation acts as a regularizer, making the model more robust to variations in the input data. This also helps the model learn AOIs in the spectrogram. 2
6) **Prefetching**: Prefetching is applied to overlap data loading with model execution, thereby improving training throughput. By preparing the next batch of data while the current batch is being processed, prefetching minimizes data loading latency and ensures that the model has a constant supply of data to work with.

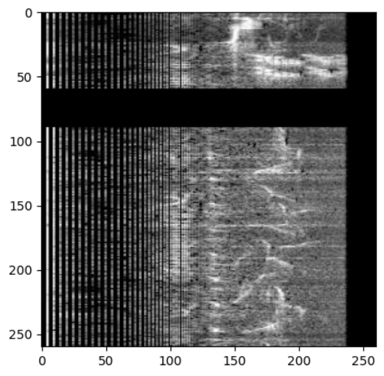
Images that look like 2 are fed into the neural network.

C. Modeling

We are using RNN [15](LSTM based, see Fig 3 4) for classification. RNNs are a class of neural networks designed to recognize patterns in sequences of data. Unlike traditional feedforward neural networks, RNNs possess an internal state that allows them to retain information about previous inputs, making them suitable for sequential data. LSTM networks, a specialized type of RNN, address the vanishing gradient problem by introducing a memory cell and gating mechanisms that regulate the flow of information. This architecture enables LSTMs to capture long-term dependencies(the reason for cropping and padding the audios at 10 seconds) and is particularly effective for tasks involving sequential data with repeating patterns. Expanding a little on the LSTM unit, it commonly contains three gates and a cell:



(a)



(b)

Fig. 2. Time-Frequency Masking. Time is in the X axis and Frequency in the Y axis

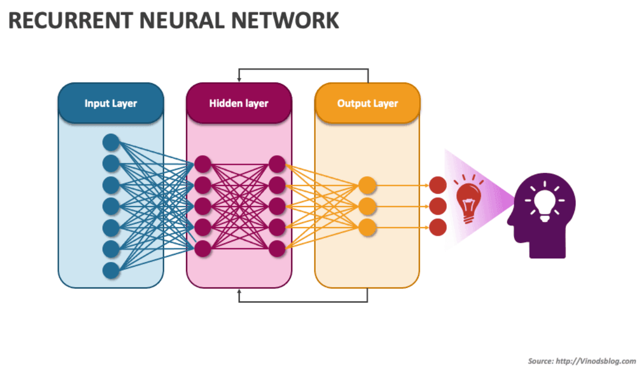


Fig. 3. A Recurrent Neural network, Source : [13]

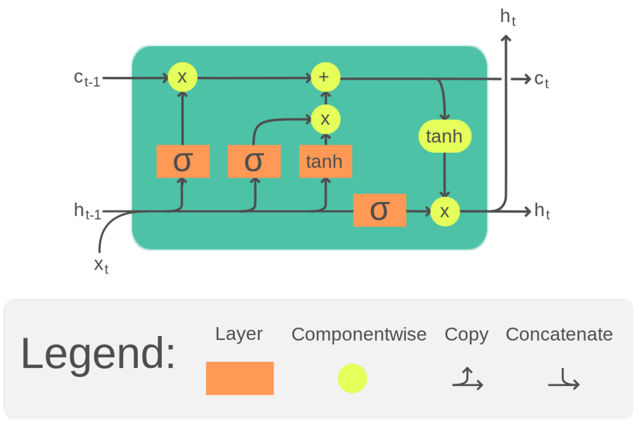


Fig. 4. A LSTM Cell, Source : [14]

- Cell
- Input Gate
- Output Gate
- Forget gate

The cell remembers values over time. The three gates regulate the flow of information into and out of the cell. Forget gates decide what information from the previous state to keep. This is done by giving a value between 0 and 1, and rounding it. If it is 0 then the information from the previous state is discarded and vice versa. Input gate works similary as forget gates and decide what information from the input to bring to the current state. Finally the output gates control what information from the current state to output by assigning a value from 0 to 1 to the information while taking in account both the previous and current states.

Mathematically we can write the cells as,

- **The forget gate**

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

  where σ is the sigmoid function, W_f is the weight matrix, b_f is the bias vector, h_{t-1} is the previous hidden state, and x_t is the current input.
- **The input gate**

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

  where i_t is the input gate's activation vector, $\tilde{c}_t$ is the candidate value vector, and W_i, W_c, b_i, b_c are the weight matrices and bias vectors.

$$\tilde{c}t = \tanh(W_c \cdot [ht - 1, x_t] + b_c)$$

- **The cell update**

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

  where $\odot$ is Hadamard product.
- **The output gate**

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

  where o_t is the output gate's activation vector, h_t is the current hidden state, and W_o and b_o are the weight matrix and bias vector for the output gate.

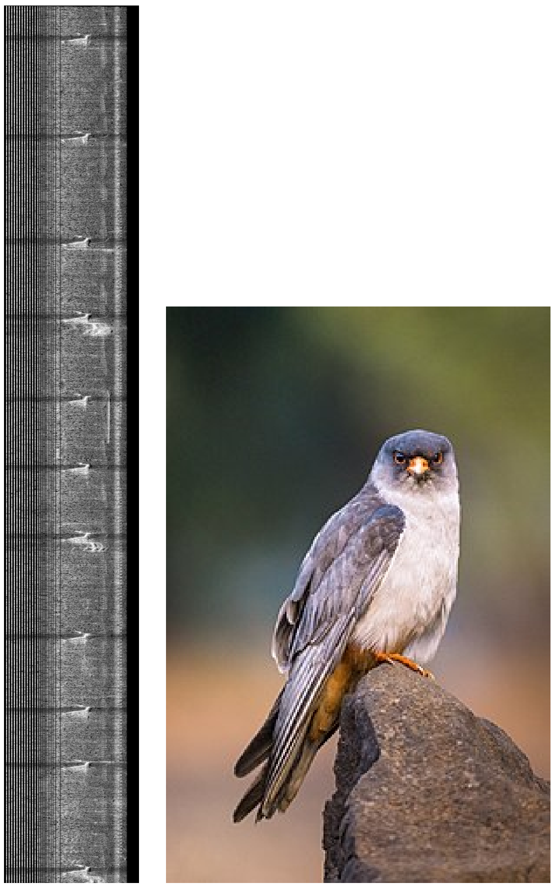
The LSTM neural network employed in this study is designed to process spectrogram data effectively. The architecture consists of the following components:

- **Input Layer:** The input layer receives spectrogram with dimensions corresponding to the height (frequency bins) and width (time frames) of the spectrogram.
- **TimeDistributed Layer:** This layer applies a Dense layer to each time step of the input separately, reshaping the input for subsequent LSTM processing.
- **Bidirectional LSTM Layer:** A bidirectional LSTM layer processes the input sequence in both forward and backward directions. This is essential to enhance capture of temporal dependencies.
- **Skip Connection:** The output of the TimeDistributed layer is merged with the output of the LSTM layer to preserve short-term features.
- **Dense and MaxPooling Layers:** These layers reduce dimensionality and extract relevant features.
- **Flatten and Dropout Layers:** The Flatten layer converts the 2D feature maps into 1D feature vectors, while the Dropout layer prevents overfitting by randomly dropping units during training.
- **Output Layer:** A Dense layer with a softmax activation function outputs the probability distribution over the classes.

The model is compiled using the Adam optimizer and sparse categorical cross entropy loss function, suitable for multi-class classification tasks. Here we use sparse categorical cross entropy instead of cross entropy on one hot encoded labels because the Jetson Nano device being used in this experiment has 4GB of RAM. Therefore one hot encoding of 264 labels eats too much memory.

Before I arrived at the above model, I tried the following approaches:

1) A deep convolution neural network 6 that takes audio as input. This neural network converted the audios to spectrograms and used a deep CNN network to classify the audios. The use of convolution neural network was hinted at by research papers [18] [19]. This model had the following issues:
   - Very high training time: Since the audios needed to be converted to spectrograms on the fly and these spectrogram were computed on the CPU. Each epoch took around 15 minutes when trained on the original audio dataset, which has 264 classes.
   - Unsatisfactory results: The trained neural network had a validation accuracy of 5% on the last epoch and test accuracy of 2% on convergence. The reason for the low accuracy might be repeating patterns are very small or subtle and the model is not able to capture at what rate they occur. Since bird sounds are taken in the wild there is also the issue of noise. One clue that the neural network has is that the areas of interest (AOIs) are timestamp in the spectrogram



(a) Mel spectrogram of Amur Falcon, Time on the Y axis and mels in the X axis

(b) Amur Falcon, Source [16]

Fig. 5. Image of spectrogram and the its respective bird

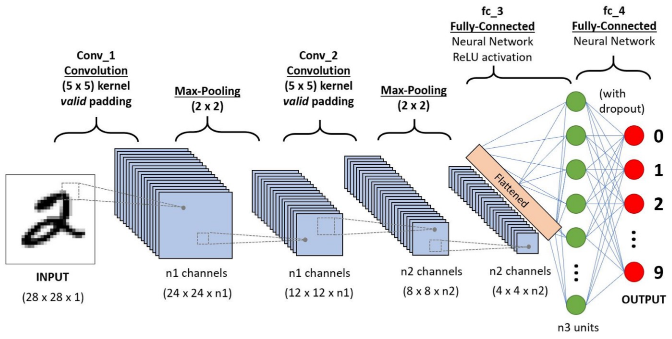


Fig. 6. Deep CNN architecture, Source [17]

where we can hear the bird in the audio / see in the spectrogram obvious spots 5(a). These occur at roughly the same interval in a audio recording. Another reason might be that because of the large number of classes the neural network is not able to capture the subtlety between the classes that have very similar looking spectrogram.

2) A Convolution Neural Network based on transfer learning, see Fig 7 on a pre-trained image model. We use a pretrained model (here I experimented with VGG16, EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB4). [18]. The spectrograms needed to be cropped to match the input spape required by the pre trained models. This method also suffered from low accuracy. On average the accuracy of transfer learning on these pretrained model only resulted in an validation accuracy of 15% on the last epoch and test accuracy of 12%. The reason according to me remains the same as the previous model. In this approach I also introduced the spectrogram dataset methodology described in section A. The average training time per epoch on this dataset was 100 seconds. Which is a major improvement over taking audio directly. Due to this I was able to make quick changes to the architecture of the model and tuning of hyper parameters. A lot of experimentation here led me to believe that CNN based architectures are not able to capture the subtlety of the data. This led me to RNN.

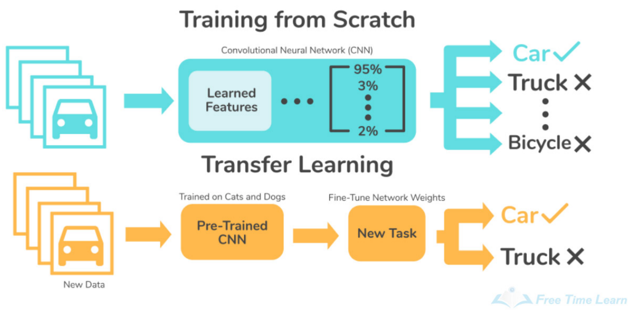


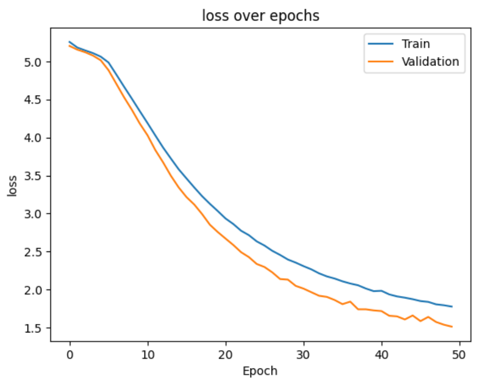
Fig. 7. Transfer Learning on a pretrained model, Source [20]

D. Training of Neural Network

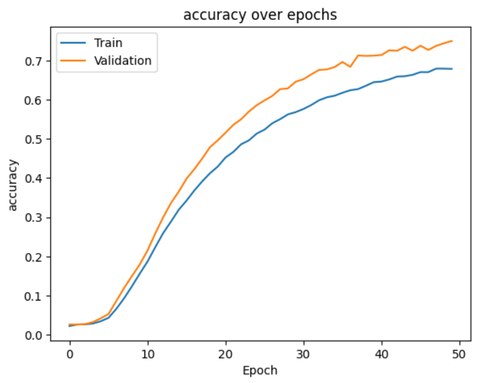
The neural network is trained on the augmented spectrogram. The model took on average 55 seconds per epoch, with an accuracy of 67 % on the training set and 74% on the testing test and 75% on validation in the last epoch. See Fig 8

E. Real time inference of bird sounds

Real-time bird sound identification involves capturing audio, preprocessing the audio data to reduce noise, converting the audio into a spectrogram, and then using our model to classify the bird sounds. This entire process is performed on the Nvidia Jetson Nano, a powerful edge computing device optimized for AI applications.



(a) Loss over Epochs



(b) Accuracy over Epochs

Fig. 8. History of training

1) **Audio Recording:** audio recording is carried out using the pyaudio library[21]. A stream is opened with the desired audio format, number of channels, sample rate, and buffer size. The system listens and captures audio data in chunks over a predefined duration. The captured audio frames are stored in a list and concatenated to form a single audio data array after recording.
2) **Noise Reduction:** Noise reduction is then applied to improve the quality of the recorded audio using the noisereduce library. Even tough our neural network is trained both noisy and non noisy data. We still use noise reduction as a precaution.
3) **Melspectrogram Generation:** The cleaned audio data is converted into a mel-spectrogram using the librosa library[22] and normalized.
4) **Inference:** The spectrogram is fed to the our neural network trained in the previous section. The model then predicts the class probabilities for the bird sound, which are printed with the bird with highest probability first.

The main loop continuously records audio every predefined interval, preprocesses it, and performs inference. This loop ensures that the system records and processes audio in real-time, waiting for a specified time before starting the next recording cycle, thus maintaining a continuous cycle of recording and inference.

Executing the entire pipeline on the Nvidia Jetson Nano leverages the device's powerful edge computing capabilities, which are optimized for AI applications. The Jetson Nano's GPU acceleration is utilized to speed up model inference and data processing tasks. This approach ensures that the real-time requirements are met without overwhelming the device's computational resources, making it well-suited for deployment in field environments where immediate identification of bird sounds is crucial.

IV. The Results and Discussion

Model	Parameters	Accuracy (%)
Deep CNN	1.5 Million	2.0
Transfer learning on VGG16	138 million	11
Transfer learning on EfficientNetB0	5.3 Million	12
RNN (LSTM)	200k	75

TABLE I
Test accuracy of different models on the test dataset

The proposed RNN (LSTM) model achieved an accuracy of 75% on the validation set for bird sound classification. This is a significant improvement over the initial CNN-based approaches that achieved low accuracies of around 5-15%. The key factors contributing to the superior performance of the RNN model are:

1) Ability to capture temporal dependencies: Bird vocalizations often exhibit repeating patterns and rhythms, which the LSTM network is well-suited to capture due to its recurrent nature and ability to model long-term dependencies.
2) Spectro-temporal feature learning: By processing the mel-spectrograms as input, the RNN model can jointly learn discriminative features across both the time and frequency domains, enabling it to better characterize the unique spectro-temporal patterns of different bird species.
3) Data augmentation and preprocessing: Techniques like time-frequency masking and cropping/padding helped increase the model's robustness and generalization capabilities, leading to improved performance on the diverse and noisy bird sound dataset.
4) Efficient data processing: The mel-spectrogram dataset creation and TensorFlow data pipeline optimizations allowed for faster training and better utilization of computational resources, enabling more effective model optimization.
5) Fast real time classification : The trained RNN has around 200000 parameters, compared to deep CNN's 1.5 Million, EfficientNetB0's 5.3 Million parameters and VGG16's 138 million, leading to faster calculations and faster results.

The results demonstrate the effectiveness of the proposed approach in addressing the challenges of bird sound classification, such as intra-class variability, background noise, and overlapping vocalizations.

V. Conclusion

This research aimed to develop a efficient bird sound classification system using deep learning techniques. The methodological approach involved preprocessing audio recordings into mel-spectrograms, which were then fed into an RNN (LSTM) model for training and inference.

The findings highlight the superiority of the RNN model over traditional CNN-based approaches for this task, achieving an accuracy of 75% on the test set. The RNN's ability to capture temporal dependencies and learn discriminative spectro-temporal features proved crucial for effectively modeling the intricate patterns present in bird vocalizations.

The theoretical contribution of this work lies in demonstrating the suitability of using Melspectrograms feeded to a RNN for classifying bird sounds. Practically, the developed system can be deployed in various applications, such as ecological monitoring, conservation efforts, and enhancing bird-watching experiences.

Future research could explore ensemble techniques, combining RNNs with other neural network architectures or hand-crafted features to further improve classification performance. The future work could also investigate in attention mechanisms and better transfer learning approaches that may lead to more efficient and generalized models. It could also explore other audio and spectrogram augmentation and feature extraction techniques.

In conclusion, this research presents a solution to classify bird in real time in their natural habitat, paving the way for more robust and accurate acoustic monitoring systems in the field of biodiversity conservation and environmental studies.

Acknowledgment

I would like to express my deepest gratitude to Professor Dr. Deepak Kumar Singh for his invaluable guidance and support throughout the course of this research. His expert insights and continuous encouragement have been instrumental in the successful completion of this study. This research would not have been possible without his mentorship and dedication.

References

[1] "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech,and Signal Processing*, 1977.

[2] A. L. McIlraith and H. C. Card, "Bird song identification using artificial neural networks and statistical analysis," in *IEEE Canadian Conference on Electrical and Computer Engineering, Engineering Innovation: Voyage of Discovery*, vol. 1, 1997, pp. 63–66.

[3] L. Neal, F. Briggs, R. Raich, and X. Z. Fern, "Time-frequency segmentation of bird song in noisy acoustic environments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011)*, 2011, pp. 2012–2015.

[4] Y. R. Leng and T. H. Dat, "Multi-label bird classification using an ensemble classifier with simple features," in *Asia Pacific Signal and Information Processing Association (APSIPA)*, 2014, pp. 1–5.

[5] D. Stowell and M. D. Plumbley, "Audio-only bird classification using unsupervised feature learning," in *Working Notes of CLEF*, 2014.

[6] S. Kahl, T. Wilhelm-Stein, H. Hussein, H. Klinck, D. Kowerko, M. Ritter, and M. Eibl, "Large-scale bird sound classification using convolutional neural networks," in *Working notes of CLEF*, 2017.

[7] V. Lostanlen, J. Salamon, and J. P. Bello, "Scattering transforms for bird song classification in noisy environments," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[8] D. Stowell, V. Lostanlen, M. Wood, and H. Davies, "Automatic acoustic detection of birds through domain adaptation and self-supervised learning," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2019.

[9] K. Prahallad, "Spectrogram, cepstrum and melfrequency analysis," Carnegie Mellon University.

[10] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition with time–frequency audio features," *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.

[11] J. Lee and I. Park, "Audio recognition using mel spectrograms and convolution neural networks," in *Report for ECE 228*. University of California San Diego, 2019.

[12] T. Authors, "Tensorflow dataset api," https://www.tensorflow.org/api_docs/python/tf/data/Dataset, 2024, version 2.16.1.

[13] paperswithcode, "Recurrent neural networks," 2024, accessed: 2024-05-23. [Online]. Available: https://almablog-media.s3.ap-south-1.amazonaws.com/image2_1_76da514c89.png

[14] Wikipedia, "Lstm cell," 2024, accessed: 2024-05-23. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/LSTM_Cell.svg/300px-LSTM_Cell.svg.png

[15] Wikipedia contributors, "Long short-term memory — Wikipedia, the free encyclopedia," 2024, [Online; accessed 31-May-2024]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1220425877

[16] Wikipedia, "Amur falcon," 2024, accessed: 2024-05-23. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/2/29/Amur_Falcon_%28male%29_%28vagrant%29.jpg/220px-Amur_Falcon_%28male%29_%28vagrant%29.jpg

[17] paperswithcode, "Convolution neural networks," 2024, accessed: 2024-05-23. [Online]. Available: https://production-media.paperswithcode.com/method_collections/cnn.jpeg

[18] A. Ashurov, Y. Zhou, L. Shi, Y. Zhao, and H. Liu, "Environmental sound classification based on transfer-learning techniques with multiple optimizers," *Electronics*, vol. 11, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/15/2279

[19] Z. S. A. F. Agnes Incze, Henrietta-Bernadett Jancso and C. Sulyok, "Bird sound recognition using a convolutional neural network," 2018.

[20] SkyEngine.AI, "What is transfer learning?" 2024. [Online]. Available: https://skyengine.ai/se/images/blog/transfer_learning.jpeg

[21] PortAudio, *PyAudio: Python Bindings for PortAudio*, retrieved from https://people.csail.mit.edu/hubert/pyaudio/. [Online]. Available: https://people.csail.mit.edu/hubert/pyaudio/

[22] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, vol. 8, 2015, pp. 18–25.

Saarthak Verma

Saarthak Verma　　Deepak Kumar Singh