KEC Conference 2025

ISSN: 3021-940X (Print) DOI: https://doi.org/10.3126/injet.v2i2.78615

CNN-Based Bird Sound Detection: A Comparative Performance Study

Gaurav Giri[1], Iza KC[2], Prajwal Khatiwada[3], Samrat Kumar Adhikari[4], Prof. Dr. Subarna Shakya[5]

[1]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, gaurovgiri@gmail.com*
[2]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, iamizakc@gmail.com*
[3]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, khatiwadaprajwal22@gmail.com*
[4]*Department of Computer and Electronics Engineering, Kantipur Engineering College, Dhapakhel, Lalitpur, Nepal, samratmetaladhikari@gmail.com*
[5]*Department of Electronics and Computer Engineering, IOE Pulchowk Campus, Pulchowk, Lalitpur, Nepal, drss@ioe.edu.np*

Abstract

The use of automated systems for biodiversity monitoring has positioned bird sound detection as a crucial technology for modern ecological research. This study introduces a deep learning framework for bird sound detection, demonstrating strong performance in challenging acoustic environments. By leveraging feature extraction techniques such as Mel spectrograms and comparing multiple convolutional neural network architectures, this approach achieves competitive results. In our experiments, the best-performing model ResNet50 achieved an accuracy of 90.75%, with a recall of 0.89, a precision of 0.91, and an F1-score of 0.90 on the test set. These results highlight the method's potential for real-time biodiversity monitoring, providing a reliable tool for ecological research, citizen science initiatives, and conservation efforts.

Keywords: Bird Sound Detection, Deep Learning, Convolutional Neural Network, Audio Processing, Mel spectrogram

1. Introduction

Birds play a crucial role in ecosystems by managing pests, promoting biodiversity, and serving as indicators of environmental health. Traditional methods of monitoring bird sounds rely on visual and manual auditory observations, which are time-consuming, labor-intensive, and limited in scope. researchers use acoustic monitoring, which allows for efficient and non-intrusive detection of bird vocalizations. Recent advancements in deep learning, particularly Convolutional Neural Networks (CNNs), has improved bird sound detection by automating the classification of audio signals. Unlike earlier approaches that required manual spectrogram analysis, CNN-based models leverage Mel Spectrogram representations to distinguish bird sounds from background noise with high accuracy. This paper explores deep learning-based techniques for bird sound detection as a binary classification task, evaluating different CNN architectures to determine the most effective approach. Accurate bird sound detection can aid conservation efforts, enhance wildlife monitoring, and provide valuable insights into environmental conditions.

2. Literature Review

The study (Gautam, et al., 2023) titled *"Audio Classifier for Automatic Identification of Endangered Bird Species of Nepal,"* investigates deep learning methods for detecting endangered bird species using audio recordings. The 2215 recordings of 41 species (38 endangered) in the collection, which was obtained from xeno-canto.org, were increased to 6733 using Gaussian noise augmentation and 10-second splits. A total of 5407 recordings were used for training, 639 for validation, and 687 for testing. Mel spectrograms and MFCCs were used to extract features and perform data augmentation to address class imbalance. A genetic algorithm was used to optimize the hyperparameters of a custom CNN and EfficientNet model. Short-time Fourier Transform, decibel scaling, and Mel filter banks were used to create Mel spectrograms, whereas Fourier

*Corresponding Author

Transform, logarithmic scaling, Mel Scaling, and Discrete Cosine Transform were used to create MFCCs. Similarly, EfficientNet used compound scaling to adjust resolution, width, and depth. The F1-scores for Model I (Mel spectrograms + EfficientNet) was 79%, Model II (Mel spectrograms + Custom CNN) was 64%, and Model III (MFCC + EfficientNet) was 72%, according to the results. The limitation of the paper is the small dataset used and the need for higher model accuracy and resilience.

The study (Lasseck, 2018), *"Acoustic Bird Detection with Deep Convolutional Neural Networks,"* examines the performance of pretrained DCNNs for bioacoustics classification on ImageNet, emphasizing the importance of preprocessing and augmentation in enhancing accuracy. Applying a shallow high-pass filter (Q = 0.707) with a 2 kHz cutoff, resampling to 22,050 Hz, and extracting 4-second audio chunks are all part of the preprocessing pipeline. These chunks are then transformed into Mel spectrograms with 310 Mel bands (160–10,300 Hz). Additional preprocessing steps include frequency filtering, power normalization to decibel units, scaling to 224×224, and converting grayscale spectrograms to RGB for ResNet compatibility. Data augmentation techniques such as jittering chunk duration, pulling chunks from random positions, applying random amplitude scaling, and introducing noise from unrelated audio recordings enhances generalization. The paper also uses other frequency-domain augmentations such piecewise time-frequency scaling, frequency shifting/stretching, and color jittering (brightness, contrast, saturation, hue). The most effective methods include adding noise from random files, applying piecewise time-frequency stretching, and using time interval dropout to enhance resilience by simulating real-world variations in bird vocalizations.

The dataset used in (Carvalho & Gomes, 2023) contains 2,730 MP3 recordings from 91 bird species (30 samples each) collected from California and Nevada, sourced from Xeno Canto. The process involves pre-processing, feature extraction, and deep learning modeling. MFCCs were extracted using python_speech_features (13 coefficients, 26 filter banks, FFT size 512), and Mel spectrograms using Librosa (FFT size 2048, hop length 512, 128 Mel bands). CNNs, particularly EfficientNet, outperformed LSTMs in classifying bird sounds, achieving 99.05% and 98.76% accuracy for 3s and 1.5s spectrograms, compared to LSTMs' 75.85% and 73.29%, showing CNNs' strength in capturing spatial and frequency patterns.

According to (Stowell, et al., 2019), recent advancements in bird audio detection leverage deep learning, particularly CNNs, which automatically extract features from spectrograms, outperforming traditional MFCCs and handcrafted spectral features. The Bird Audio Detection challenge evaluated models like CNNs, RNNs, and SVMs, with CNNs achieving the highest AUC scores (up to 95% in matched conditions and 88% in mismatched ones). Despite strong performance, challenges remain in handling background noise, faint calls, and calibration inconsistencies, highlighting the need for further research in model generalization and reliability.

The paper (Grill & Schlüter, 2017) tackled bird vocalization detection using two CNNs for the Bird Audio Detection Challenge, achieving 89% AUC. They trained on freefield1010 and Warblrb datasets but faced domain adaptation issues when testing on unseen TREE recordings from Chernobyl. Both models processed mel spectrograms but differed in architecture: **bulbul** used a 14s receptive field with convolutional-pooling layers, while **sparrow** applied multiple-instance learning on 1.5s windows. Data augmentation (time/pitch shifting, noise mixing) improved robustness. Despite domain mismatches (Pearson: 0.40), **bulbul** (88.76% AUC) slightly outperformed **sparrow** (88.41%), with their ensemble reaching 89.68%. The study emphasized domain-invariant augmentation and better labeling for bioacoustics recognition.

The study (Nanni, et al., 2021), *"An Ensemble of Convolutional Neural Networks for Audio Classification,"* investigates how to improve audio classification performance utilizing CNN-based classification with different architectures, data augmentation methods, and audio signal formats. It assesses three datasets with distinct categorization problems: ESC-50, CAT, and BirdZ. The approach creates thirty-five ensemble subtypes by training five convolutional neural networks (CNNs) using four distinct audio representations and six data augmentation strategies. The augmentation techniques include frequency masking, random time shift, and brief spectrogram augmentation, while the audio representations include Waveform Similarity OverLap Add (WSOLA), Discrete Gabor Transform (DGT), and Phase Vocoder. The CNN models are pre-trained and adjusted with enhanced datasets to increase the accuracy. The ensemble technique achieves 97% accuracy on

BIRDZ, 90.51% on CAT, and 88.65% on ESC-50, demonstrating its superior performance over individual networks. According to the study, DGT is the most efficient signal representation, and VGG16 and VGG19 are the top-performing CNN architectures. Despite encouraging results, challenges include high computational costs for training ensembles and performance variability across different augmentation strategies.

In (Incze, et al., 2018), to categorize bird cries, a CNN-based system was created utilizing spectrograms produced from recordings taken from the popular bird song archive Xeno-canto. The study tested several hyperparameters, such as the number of bird species and spectrogram color schemes and refined a pre-trained MobileNet model. The results showed that classification accuracy improved when using a colourmap matching the model's pre-training data. However, the study found that the system's performance was limited when handling more classes, suggesting it is only practical for classification tasks with a smaller scope. According to the experiments, the model did well for two classes but experienced a sharp decline in accuracy as the number of classes rose, dropping below 40% for ten classes. The higher average accuracy of 7.4% for spectrograms using the Jet colourmap compared to grayscale is likely due to MobileNet's pre-training on color images. The Jet color map was a better option for more significant classifications because the accuracy gap grew as the number of classes increased.

In (Hu, et al., 2023), a feature fusion network (MFF-ScSEnet) was developed to combine Sinc-spectrograms, which capture timbral properties, with Mel-spectrograms, which concentrate low-frequency aspects, to address information loss during spectrogram creation. A ResNet18 backbone supplemented with the ScSEnet attention module was utilized to process the fused features to increase sound ripple information and decrease noise interference. It outperformed recent birdsong recognition techniques with evaluations of the public datasets (Urbansound8K and Birdsdata) and the self-built Huabei dataset, with accuracies of 96.28%, 98.34%, and 96.66%, respectively.

3. Methodology

This paper explores the generation of Mel spectrograms images from audio files as a preprocessing step for classification. Different convolutional neural network (CNN) architectures are used to train various models, all with the same set of hyperparameters to ensure a fair comparison. These trained models classify whether a given audio sample contains bird sounds. Finally, the paper compares their performance on a standardized test set to identify the most effective architecture for this task.

3.1. Dataset Description

The dataset used for the Bird Sound Detection Model includes two primary sources:

Field recordings, worldwide *(Papers with Code freefield1010, n.d.)****:***

The collection consists of 7,690 excerpts from field recordings worldwide, gathered by the FreeSound project then standardized for research. The dataset is diverse in location and environment and has been annotated for the presence or absence of birds. The distribution of the dataset is shown in Figure 1.

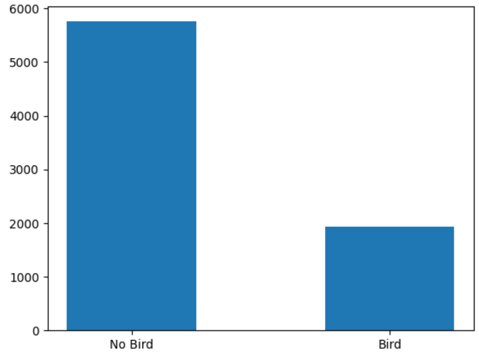


Figure 1. Audio distribution of freefield1010 dataset

Crowdsourced dataset, UK** (Papers with Code warblrb10k, n.d.)**:

This dataset includes 8,000 smartphone audio recordings around the UK, crowdsourced by users of Warblrb, the bird recognition app. The audio covers a wide distribution of UK locations and environments and includes weather noise, traffic noise, human speech, and even 18human bird imitations. The distribution of this dataset is shown in Figure 2.

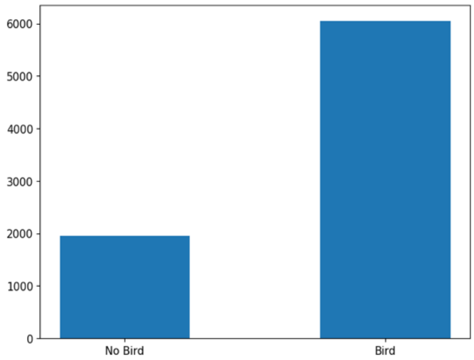


Figure 2. Audio distribution of warblrb10k dataset

Instead of using the datasets separately, we merged them to create a single dataset comprising 15,690 audio recordings - 7,710 without bird sounds and 7,980 with bird sounds. The merged dataset was then partitioned into training, testing, and validation sets in a 75:15:10 ratio It ensures a comprehensive and balanced dataset for model training and evaluation.

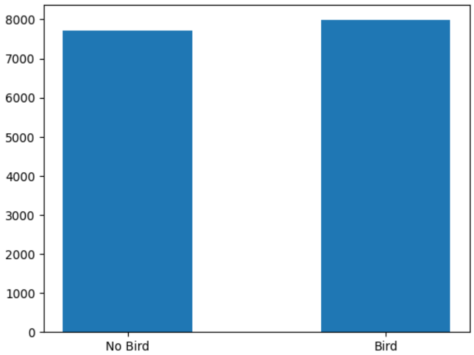


Figure 3. Audio distribution of each class after merging the datasets

3.2. Data Transformation

To enhance the robustness and generalization of the bird sound detection model, a series of transformations were applied to both the raw audio signals and their corresponding Mel spectrograms. These transformations include:

• Add Gaussian Noise (Raw Audio):

This transformation adds Gaussian noise to the raw audio signal. The noise is generated with a specified mean and standard deviation, and it is added to the audio signal with a certain probability. It helps in making the model robust to noisy environments.

$$\textbf{noise} = \mathbf{N}\ (\mu, \sigma^2) \qquad \text{(Equation 1)}$$

• Random Volume Scaling (Raw Audio):

This transformation scales the volume of the raw audio signal by a random factor within a specified range. The scaling is applied with a certain probability, making the model resilient to variations in recording levels.

$$\textbf{audio = audio} \times \textbf{gain} \qquad \text{(Equation 2)}$$

• Time Stretch (Raw Audio):

This transformation stretches or compresses the time axis of the raw audio sig-nal by a random factor within a specified range. The transformed audio is then converted into a Mel spectrogram, which helps the model handle variations in speed.

$$\textbf{audio = TimeStretch (audio, rate)} \qquad \text{(Equation 3)}$$

• Frequency Masking (Mel spectrograms):

This transformation masks a portion of the frequency bins in the MelSpectro-gram. The maximum number of frequency bins to be masked is specified, and the masking is applied with a certain probability. It encourages the model to focus on different frequency components.

$$\textbf{Melspec [f: f + max mask] = 0} \qquad \text{(Equation 4)}$$

• Time Masking (Mel spectrograms):

This transformation masks a portion of the time frames in the Mel spectrograms. The maximum number of time frames to be masked is specified, and the masking is applied with a certain probability. It helps the model to focus on various temporal features.

$$\textbf{Melspec [:, t: t + max mask] = 0} \qquad \text{(Equation 5)}$$

• Random Content Mixing (Mel spectrograms):

This transformation mixes the Mel spectrograms with another randomly selected noise Mel spectrograms. The mixing ratio is chosen randomly within a specified range, and the mixing is applied with a certain probability. This aids in making the model robust to background noises.

$$\textbf{Melspec = (1} - \textbf{mix ratio)} \times \textbf{Melspec + mix ratio} \times \textbf{noise spec} \qquad \text{(Equation 6)}$$

• Time Interval Dropout (Mel spectrograms):

This transformation randomly drops intervals of time frames in the MelSpectro-gram. The number of intervals and their width are randomly selected within specified ranges and applied with a certain probability. This approach helps the model handle missing or corrupted audio segments more effectively.

$$\textbf{Melspec[:, t: t + width] = 0} \qquad \text{(Equation 7)}$$

3.3. Feature Extraction

The Mel spectrogram transforms raw audio signals into a time-frequency image that captures the essential spectral characteristics required for bird sound detection. As demonstrated by (Lasseck, 2018) and (Zhang, et al., 2019), this approach efficiently distinguishes recordings containing bird vocalizations from those without. For a comprehensive explanation of the computation, please refer to the 3.4

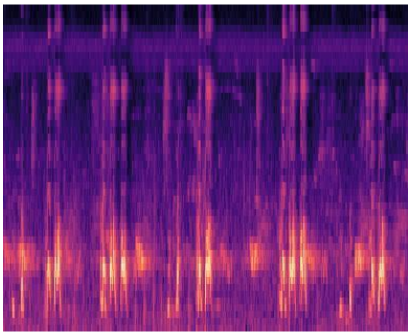


Figure 4. Mel spectrogram image for sample audio

3.4. Model Selection

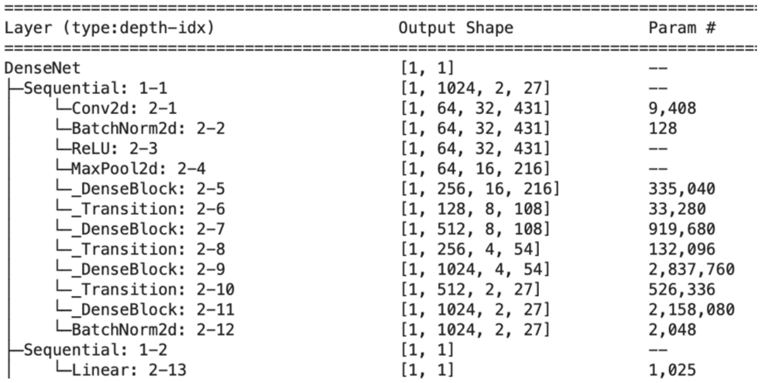
CNN-based architectures are considered nearly perfect for processing Mel spectrograms in bird sound classification because they efficiently capture both local and global acoustic features, automatically learning hierarchical representations that distinguish subtle patterns in bird calls even under noisy conditions by converting raw audio into Mel spectrograms.

In (Carvalho & Gomes, 2023), the study demonstrated the superior performance of fine-tuned CNN models over RNNs like LSTMs, achieving near-perfect classification accuracy with Mel spectrograms for bird species identification. The study utilized a comprehensive dataset with 91 species and employed feature extraction methods like MFCCs and Mel spectrograms, yielding 99.05% and 98.76% accuracy for 3-second and 1.5-second spectrograms, respectively.

These networks effectively reveal key elements such as pitch, timbre, and temporal modulations, which are critical for accurately identifying species, and numerous studies have demonstrated that fine-tuning pre-trained CNN models on such spectrogram images can yield near-perfect classification performance in large-scale challenges, underscoring their robustness and reliability in ecological monitoring and biodiversity conservation.

3.4.1. DenseNet121

DenseNet121 is a robust neural network architecture that excels at image identification and is distinguished by its dense connectivity layout. DenseNet121 connects every layer to every other layer in a feed-forward manner, in contrast to standard networks, which only connect each layer to the layer after it. Dense connectivity improves accuracy and training efficiency by encouraging feature reuse, improving gradient flow, and reducing the vanishing gradient problem. (Huang, et al., 2017).

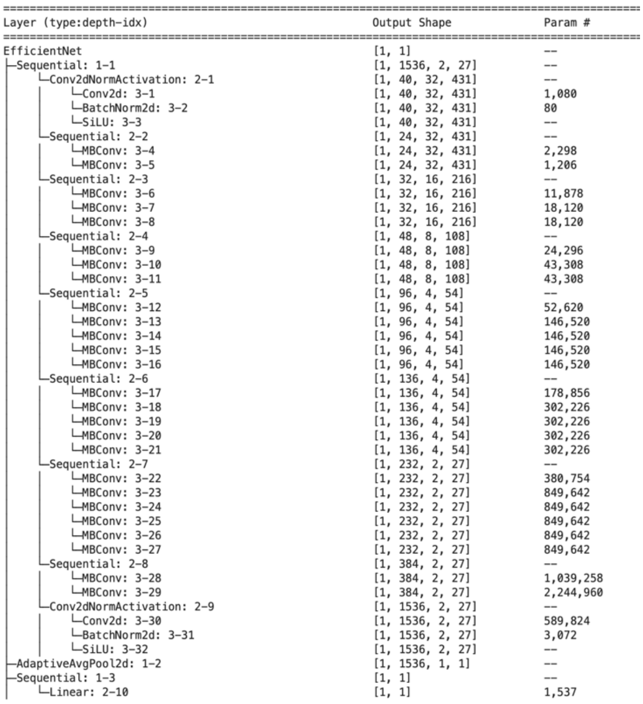


```
===============================================================================
Layer (type:depth-idx)                   Output Shape              Param #
===============================================================================
DenseNet                                 [1, 1]                    --
├─Sequential: 1-1                        [1, 1024, 2, 27]          --
│    └─Conv2d: 2-1                       [1, 64, 32, 431]          9,408
│    └─BatchNorm2d: 2-2                  [1, 64, 32, 431]          128
│    └─ReLU: 2-3                         [1, 64, 32, 431]          --
│    └─MaxPool2d: 2-4                    [1, 64, 16, 216]          --
│    └─_DenseBlock: 2-5                  [1, 256, 16, 216]         335,040
│    └─_Transition: 2-6                  [1, 128, 8, 108]          33,280
│    └─_DenseBlock: 2-7                  [1, 512, 8, 108]          919,680
│    └─_Transition: 2-8                  [1, 256, 4, 54]           132,096
│    └─_DenseBlock: 2-9                  [1, 1024, 4, 54]          2,837,760
│    └─_Transition: 2-10                 [1, 512, 2, 27]           526,336
│    └─_DenseBlock: 2-11                 [1, 1024, 2, 27]          2,158,080
│    └─BatchNorm2d: 2-12                 [1, 1024, 2, 27]          2,048
├─Sequential: 1-2                        [1, 1]                    --
│    └─Linear: 2-13                      [1, 1]                    1,025
```

Figure 5. Detailed architecture of DenseNet121

3.4.2. EfficientNetB3

EfficientNet is a convolutional neural network built upon a concept called "compound scaling.” The long-standing trade-off between model size, accuracy, and computational efficiency is addressed by this idea. Compound scaling aims to scale the three key components of a neural network: resolution, depth, and width (Tan & Le, 2023).

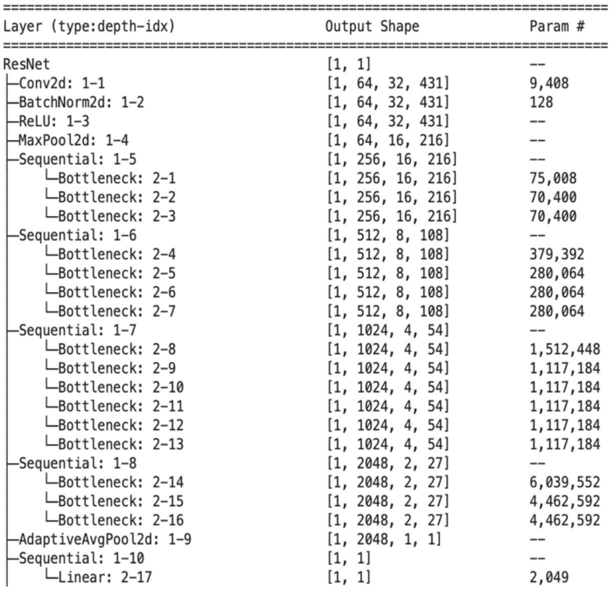


```
==========================================================================================
Layer (type:depth-idx)                                Output Shape            Param #
==========================================================================================
EfficientNet                                          [1, 1]                  --
├─Sequential: 1-1                                     [1, 1536, 2, 27]        --
│    └─Conv2dNormActivation: 2-1                      [1, 40, 32, 431]        --
│    │    └─Conv2d: 3-1                               [1, 40, 32, 431]        1,080
│    │    └─BatchNorm2d: 3-2                          [1, 40, 32, 431]        80
│    │    └─SiLU: 3-3                                 [1, 40, 32, 431]        --
│    └─Sequential: 2-2                                [1, 24, 32, 431]        --
│    │    └─MBConv: 3-4                               [1, 24, 32, 431]        2,298
│    │    └─MBConv: 3-5                               [1, 24, 32, 431]        1,206
│    └─Sequential: 2-3                                [1, 32, 16, 216]        --
│    │    └─MBConv: 3-6                               [1, 32, 16, 216]        11,878
│    │    └─MBConv: 3-7                               [1, 32, 16, 216]        18,120
│    │    └─MBConv: 3-8                               [1, 32, 16, 216]        18,120
│    └─Sequential: 2-4                                [1, 48, 8, 108]         --
│    │    └─MBConv: 3-9                               [1, 48, 8, 108]         24,296
│    │    └─MBConv: 3-10                              [1, 48, 8, 108]         43,308
│    │    └─MBConv: 3-11                              [1, 48, 8, 108]         43,308
│    └─Sequential: 2-5                                [1, 96, 4, 54]          --
│    │    └─MBConv: 3-12                              [1, 96, 4, 54]          52,620
│    │    └─MBConv: 3-13                              [1, 96, 4, 54]          146,520
│    │    └─MBConv: 3-14                              [1, 96, 4, 54]          146,520
│    │    └─MBConv: 3-15                              [1, 96, 4, 54]          146,520
│    │    └─MBConv: 3-16                              [1, 96, 4, 54]          146,520
│    └─Sequential: 2-6                                [1, 136, 4, 54]         --
│    │    └─MBConv: 3-17                              [1, 136, 4, 54]         178,856
│    │    └─MBConv: 3-18                              [1, 136, 4, 54]         302,226
│    │    └─MBConv: 3-19                              [1, 136, 4, 54]         302,226
│    │    └─MBConv: 3-20                              [1, 136, 4, 54]         302,226
│    │    └─MBConv: 3-21                              [1, 136, 4, 54]         302,226
│    └─Sequential: 2-7                                [1, 232, 2, 27]         --
│    │    └─MBConv: 3-22                              [1, 232, 2, 27]         380,754
│    │    └─MBConv: 3-23                              [1, 232, 2, 27]         849,642
│    │    └─MBConv: 3-24                              [1, 232, 2, 27]         849,642
│    │    └─MBConv: 3-25                              [1, 232, 2, 27]         849,642
│    │    └─MBConv: 3-26                              [1, 232, 2, 27]         849,642
│    │    └─MBConv: 3-27                              [1, 232, 2, 27]         849,642
│    └─Sequential: 2-8                                [1, 384, 2, 27]         --
│    │    └─MBConv: 3-28                              [1, 384, 2, 27]         1,039,258
│    │    └─MBConv: 3-29                              [1, 384, 2, 27]         2,244,960
│    └─Conv2dNormActivation: 2-9                      [1, 1536, 2, 27]        --
│    │    └─Conv2d: 3-30                              [1, 1536, 2, 27]        589,824
│    │    └─BatchNorm2d: 3-31                         [1, 1536, 2, 27]        3,072
│    │    └─SiLU: 3-32                                [1, 1536, 2, 27]        --
├─AdaptiveAvgPool2d: 1-2                              [1, 1536, 1, 1]         --
├─Sequential: 1-3                                     [1, 1]                  --
│    └─Linear: 2-10                                   [1, 1]                  1,537
```

Figure 6. Detailed architecture of EfficientNetB3

3.4.3. ResNet50

ResNet50, a neural network developed by Microsoft, is specifically made for uses like image recognition. 48 convolutional layers, one max pooling layer, and one average pooling layer make up this 50-layer convolutional neural network. It is a kind of artificial neural network (ANN) that builds networks by stacking residual blocks (Mascarenhas & Agarwal, 2021).

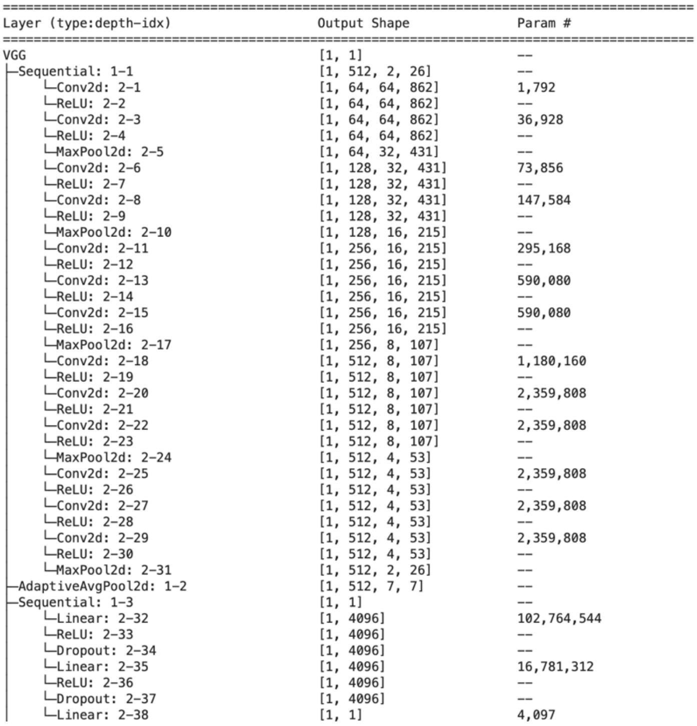


```
===========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
===========================================================================
ResNet                                   [1, 1]                    --
├─Conv2d: 1-1                            [1, 64, 32, 431]          9,408
├─BatchNorm2d: 1-2                       [1, 64, 32, 431]          128
├─ReLU: 1-3                              [1, 64, 32, 431]          --
├─MaxPool2d: 1-4                         [1, 64, 16, 216]          --
├─Sequential: 1-5                        [1, 256, 16, 216]         --
│    └─Bottleneck: 2-1                   [1, 256, 16, 216]         75,008
│    └─Bottleneck: 2-2                   [1, 256, 16, 216]         70,400
│    └─Bottleneck: 2-3                   [1, 256, 16, 216]         70,400
├─Sequential: 1-6                        [1, 512, 8, 108]          --
│    └─Bottleneck: 2-4                   [1, 512, 8, 108]          379,392
│    └─Bottleneck: 2-5                   [1, 512, 8, 108]          280,064
│    └─Bottleneck: 2-6                   [1, 512, 8, 108]          280,064
│    └─Bottleneck: 2-7                   [1, 512, 8, 108]          280,064
├─Sequential: 1-7                        [1, 1024, 4, 54]          --
│    └─Bottleneck: 2-8                   [1, 1024, 4, 54]          1,512,448
│    └─Bottleneck: 2-9                   [1, 1024, 4, 54]          1,117,184
│    └─Bottleneck: 2-10                  [1, 1024, 4, 54]          1,117,184
│    └─Bottleneck: 2-11                  [1, 1024, 4, 54]          1,117,184
│    └─Bottleneck: 2-12                  [1, 1024, 4, 54]          1,117,184
│    └─Bottleneck: 2-13                  [1, 1024, 4, 54]          1,117,184
├─Sequential: 1-8                        [1, 2048, 2, 27]          --
│    └─Bottleneck: 2-14                  [1, 2048, 2, 27]          6,039,552
│    └─Bottleneck: 2-15                  [1, 2048, 2, 27]          4,462,592
│    └─Bottleneck: 2-16                  [1, 2048, 2, 27]          4,462,592
├─AdaptiveAvgPool2d: 1-9                 [1, 2048, 1, 1]           --
├─Sequential: 1-10                       [1, 1]                    --
│    └─Linear: 2-17                      [1, 1]                    2,049
```

Figure 7. Detailed architecture of ResNet50

3.4.4. VGG16

A 16-layer deep convolutional neural network (CNN) architecture called VGG16 was put forth by the University of Oxford's Visual Geometry Group. Three fully connected layers and thirteen convolutional layers make up the network. The RGB photos have an input size of 224x224x3. To preserve spatial dimensions, the convolutional layers employ 3x3 filters with padding and a stride of 1. There are over 138 million parameters in the network overall. The simplicity and efficacy of the VGG16 architecture are well known (Tammina, 2019).



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
VGG                                      [1, 1]                    --
├─Sequential: 1-1                        [1, 512, 2, 26]           --
│    └─Conv2d: 2-1                       [1, 64, 64, 862]          1,792
│    └─ReLU: 2-2                         [1, 64, 64, 862]          --
│    └─Conv2d: 2-3                       [1, 64, 64, 862]          36,928
│    └─ReLU: 2-4                         [1, 64, 64, 862]          --
│    └─MaxPool2d: 2-5                    [1, 64, 32, 431]          --
│    └─Conv2d: 2-6                       [1, 128, 32, 431]         73,856
│    └─ReLU: 2-7                         [1, 128, 32, 431]         --
│    └─Conv2d: 2-8                       [1, 128, 32, 431]         147,584
│    └─ReLU: 2-9                         [1, 128, 32, 431]         --
│    └─MaxPool2d: 2-10                   [1, 128, 16, 215]         --
│    └─Conv2d: 2-11                      [1, 256, 16, 215]         295,168
│    └─ReLU: 2-12                        [1, 256, 16, 215]         --
│    └─Conv2d: 2-13                      [1, 256, 16, 215]         590,080
│    └─ReLU: 2-14                        [1, 256, 16, 215]         --
│    └─Conv2d: 2-15                      [1, 256, 16, 215]         590,080
│    └─ReLU: 2-16                        [1, 256, 16, 215]         --
│    └─MaxPool2d: 2-17                   [1, 256, 8, 107]          --
│    └─Conv2d: 2-18                      [1, 512, 8, 107]          1,180,160
│    └─ReLU: 2-19                        [1, 512, 8, 107]          --
│    └─Conv2d: 2-20                      [1, 512, 8, 107]          2,359,808
│    └─ReLU: 2-21                        [1, 512, 8, 107]          --
│    └─Conv2d: 2-22                      [1, 512, 8, 107]          2,359,808
│    └─ReLU: 2-23                        [1, 512, 8, 107]          --
│    └─MaxPool2d: 2-24                   [1, 512, 4, 53]           --
│    └─Conv2d: 2-25                      [1, 512, 4, 53]           2,359,808
│    └─ReLU: 2-26                        [1, 512, 4, 53]           --
│    └─Conv2d: 2-27                      [1, 512, 4, 53]           2,359,808
│    └─ReLU: 2-28                        [1, 512, 4, 53]           --
│    └─Conv2d: 2-29                      [1, 512, 4, 53]           2,359,808
│    └─ReLU: 2-30                        [1, 512, 4, 53]           --
│    └─MaxPool2d: 2-31                   [1, 512, 2, 26]           --
├─AdaptiveAvgPool2d: 1-2                 [1, 512, 7, 7]            --
├─Sequential: 1-3                        [1, 1]                    --
│    └─Linear: 2-32                      [1, 4096]                 102,764,544
│    └─ReLU: 2-33                        [1, 4096]                 --
│    └─Dropout: 2-34                     [1, 4096]                 --
│    └─Linear: 2-35                      [1, 4096]                 16,781,312
│    └─ReLU: 2-36                        [1, 4096]                 --
│    └─Dropout: 2-37                     [1, 4096]                 --
│    └─Linear: 2-38                      [1, 1]                    4,097
```

Figure 8. Detailed architecture of VGG16

3.5. Evaluation Metrics

The standardized training dataset is used to build and train the system model. The performance of the model is then confirmed by testing it with the testing dataset. The system's performance is measured using metrics such as recall, F1 score, accuracy, and precision. A classification model's efficacy is assessed using a table known as the confusion matrix. It provides a matrix-formatted summary of the model's predictions based on a dataset.

4. Experimental results

Table 1. Hyperparameters used

Parameter	Value
Epoch	20
Batch size	32
Optimizer	AdamW
Learning rate	0.001
Sample Rate	44100 Hz
Duration	10 seconds
No of Mels	64
No of Samples	Sample rate * Duration
No of FFT	1024
Hop Length	No of FFT / 2
Freq Min	1000 Hz
Freq Max	8000 Hz
Weight Decay	$1e^{-5}$
Momentum	0.9

The chosen hyperparameters are optimized for efficient audio signal processing and deep learning model training.

4.1. DenseNet121

The accuracy of the model for testing data: 87.79%
The loss of the model for testing data: 0.3276

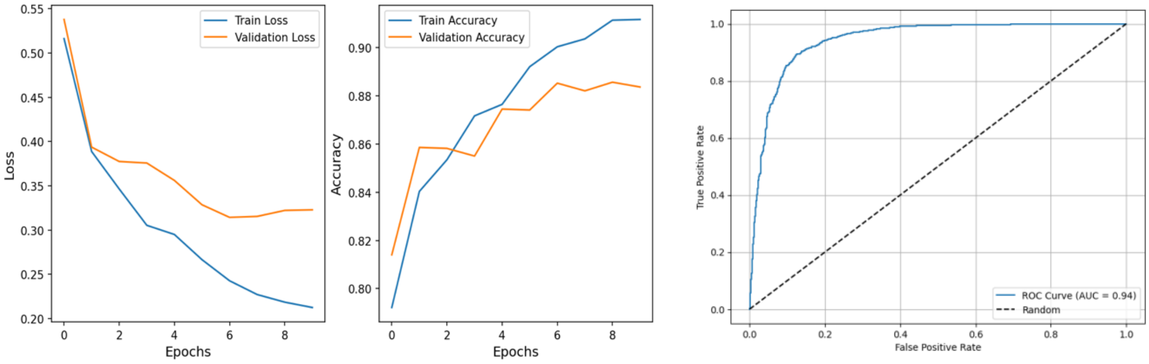


Figure 9. Cross Entropy Loss, Classification Accuracy and ROC Curve for DenseNet121

4.2. EfficientNetB3

The accuracy of the model for testing data: 87.50%
The loss of the model for testing data: 0.3209

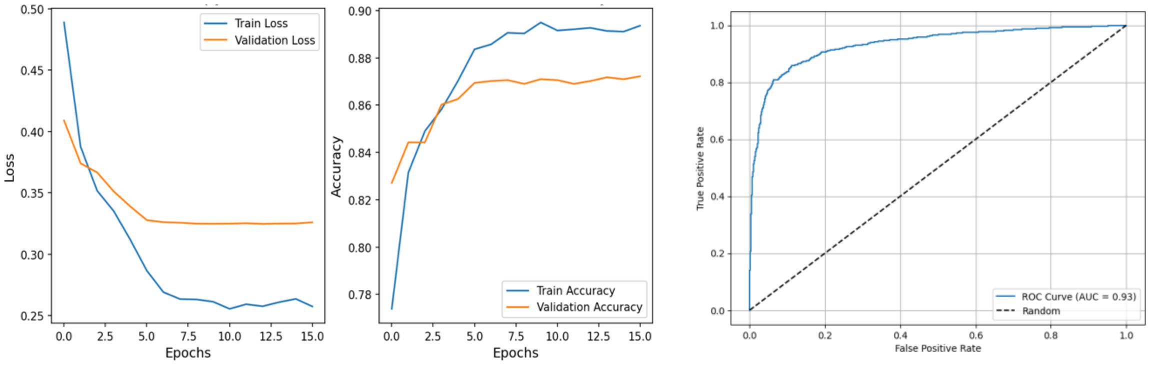


Figure 10. Cross Entropy Loss, Classification Accuracy and ROC Curve for EfficientNetB3

4.3. ResNet50

The accuracy of the model for testing data: 90.75%
The loss of the model for testing data: 0.284

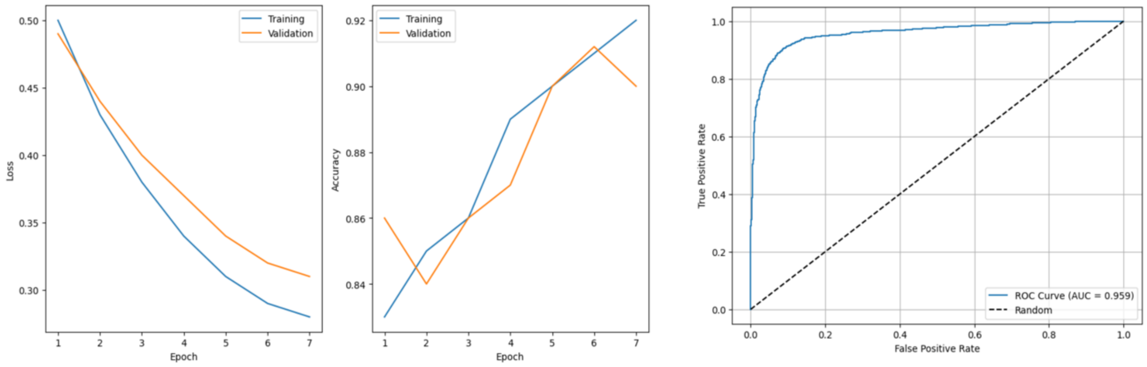


Figure 11. Cross Entropy Loss, Classification Accuracy and ROC Curve for ResNet50

4.4. VGG16

The accuracy of the model for testing data: 87.85%
The loss of the model for testing data: 0.3175

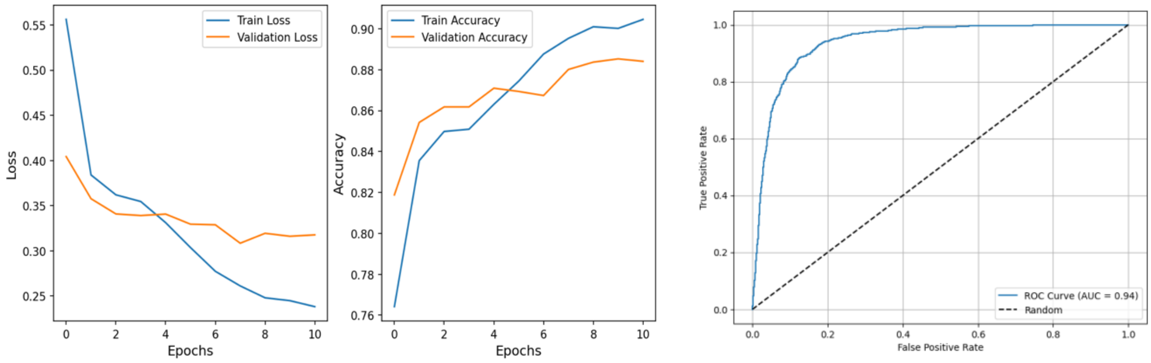


Figure 12. Cross Entropy Loss, Classification Accuracy and ROC Curve for VGG16

4.5. Modelcomparison

Table 2. Model performance comparison on test set

Model	Accuracy (%)	Recall	Precision	F1-score	AUC ROC value
DenseNet121	87.79%	0.88	0.88	0.88	0.940
EfficientNetB3	87.50%	0.87	0.88	0.88	0.930
ResNet50	**90.74%**	**0.89**	**0.91**	**0.90**	**0.959**
VGG16	87.85%	0.88	0.88	0.88	0.940

The table above presents the performance metrics of various convolutional neural network (CNN) models on a classification task. Each model, including DenseNet121, EfficientNetB3, ResNet50 and VGG16, is evaluated based on accuracy, recall, precision, and F1-score.

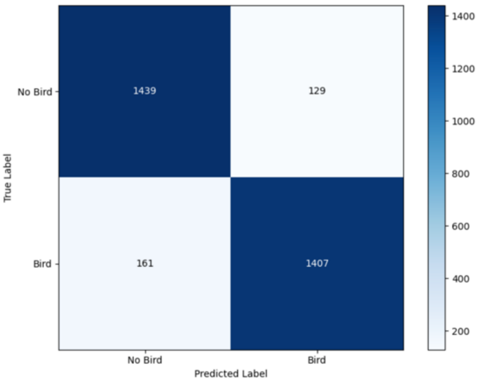


Figure 13. Confusion Matrix for ResNet50 on Testing Dataset

5. Conclusion and Future Enhancement

In conclusion, this study demonstrates that deep learning-based approaches, particularly CNN architectures utilizing Mel spectrogram features, provide robust and reliable performance for bird sound detection and classification. Among the evaluated models DenseNet121, EfficientNetB3, and VGG16, ResNet50 emerges as the best-performing architecture, achieving the highest accuracy, recall, precision, and F1-score.

ResNet50's superior performance can be attributed to its deep residual learning framework, which enables efficient training of very deep networks by mitigating vanishing gradient issues. The skip connections in ResNet50 allow the model to capture intricate time–frequency patterns in Mel spectrogram images more effectively than shallower architectures. This advantage is particularly significant in bird sound detection, where subtle frequency variations and temporal structures distinguish real bird calls from background noise or artificial sounds. Additionally, compared to lighter models like ResNet50 demonstrated better feature extraction capacity, while outperforming deeper architectures like DenseNet121 in terms of training stability and convergence.

The novelty of this study lies in its comparative evaluation of multiple CNN architectures specifically for bird sound detection, offering insights into how different model complexities impact classification performance. By converting 10-second audio recordings into Mel spectrogram images, this research reinforces the effectiveness of image-based deep learning techniques for bioacoustics analysis. The findings underscore ResNet50's potential as a valuable tool for enhancing avian monitoring and biodiversity conservation, particularly in regions where precise and automated ecological data is critical.

Looking ahead, future enhancements could focus on several key areas to further improve model performance and applicability. Integrating additional data augmentation techniques, such as incorporating realistic environmental noise profiles, time-frequency stretching, or adversarial training, could bolster the model's resilience against diverse recording conditions. Moreover, exploring hybrid architectures that combine the powerful feature extraction of ResNet50 with temporal modeling layers like LSTM or GRU may better capture sequential dependencies in bird calls, leading to even more accurate detection. Finally, optimizing and compressing the model for deployment on edge devices would enable real-time monitoring in the field, facilitating its application in citizen science projects and long-term ecological studies.

Acknowledgment

The authors would like to express their profound gratitude to the Department of Computer and Electronics Engineering at Kantipur Engineering College for their steadfast support and resources, which have proven indispensable throughout this research project. The authors are grateful to everyone who helped make this research possible.

References

Nanni, L., Maguolo, G., Brahnam, S. & Paci, M. (2021). An ensemble of convolutional neural networks for audio classification. Applied Sciences, 11(13), 5796. MDPI AG, Basel, Switzerland.

Gautam, R., Khatiwada, B., Subedi, B.P., Duwal, N. & Dahal, K.C. (2023). Audio classifier for automatic identification of endangered bird species of Nepal. In Proceedings of the 13th IOE Graduate Conference. Institute of Engineering, Tribhuvan University, Kathmandu, Nepal.

Lasseck, M. (2018). Acoustic bird detection with deep convolutional neural networks. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018), pp. 143–147. Museum für Naturkunde, Berlin, Germany.

Incze, Á., Jancsó, H.-B., Szilágyi, Z. & Sulyok, C. (2018). Bird sound recognition using a convolutional neural network. In Proceedings of the 16th IEEE International Symposium on Intelligent Systems and Informatics (SISY 2018), pp. 295–300. IEEE, Subotica, Serbia.

Hu, S., Chu, Y., Wen, Z., Zhou, G., Sun, Y. & Chen, A. (2023). Deep learning bird song recognition based on MFF-ScSEnet. Ecological Indicators, 110844. Elsevier BV, Amsterdam, Netherlands.

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K.Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), pp. 4700–4708. IEEE, Honolulu, HI, USA.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), pp. 4510–4520. IEEE, Salt Lake City, UT, USA.

Zhang, B., Leitner, J. & Thornton, S. (2019). Audio recognition using Mel spectrograms and convolutional neural networks. University of California, San Diego.

Tan, M. & Le, Q.V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Proceedings of the 36th International Conference on Machine Learning (ICML 2019), pp. 6105–6114. PMLR, Long Beach, CA, USA.

Mascarenhas, S. & Agarwal, M. (2021). A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for image classification. In Proceedings of the 2021 International Conference on Disruptive *Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, pp. 96–99. IEEE, Bengaluru, India.

Tammina, S. (2019). Transfer learning using VGG-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications*, 9(10), 9420. IJSRP Inc., Delhi, India.

Carvalho, S. & Gomes, E.F. (2023). Automatic classification of bird sounds: using MFCC and Mel spectrogram features with deep learning. *Vietnam Journal of Computer Science*, 10, 39–54. World Scientific Publishing, Singapore.

Stowell, D., Wood, M., Pamuła, H., Stylianou, Y. & Glotin, H. (2019). Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge. *Methods in Ecology and Evolution*, 10(3), 368–380. Wiley, Hoboken, NJ, USA.

Grill, T. & Schlüter, J. (2017). Two convolutional neural networks for bird detection in audio signals. In *Proceedings of the 25th European Signal Processing Conference (EUSIPCO 2017)*, pp. 1764–1768. IEEE, Kos, Greece.

Papers with Code freefield1010, n.d. *freefield1010.* [Online]
Available at: https://paperswithcode.com/dataset/freefield1010 [Accessed 12 4 2025].

Papers with Code warblrb10k, n.d. *warblrb10k.* [Online]
Available at: https://paperswithcode.com/dataset/warblrb10k [Accessed 12 4 2025].