Bird Sound Classification Using MEL Spectrograms

CONTRIBUTORS:

- WALI SIDDIQUI
- ANURAG SURVE
- WILONA NGUYEN

GITHUB REPO LINK

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Roadmap

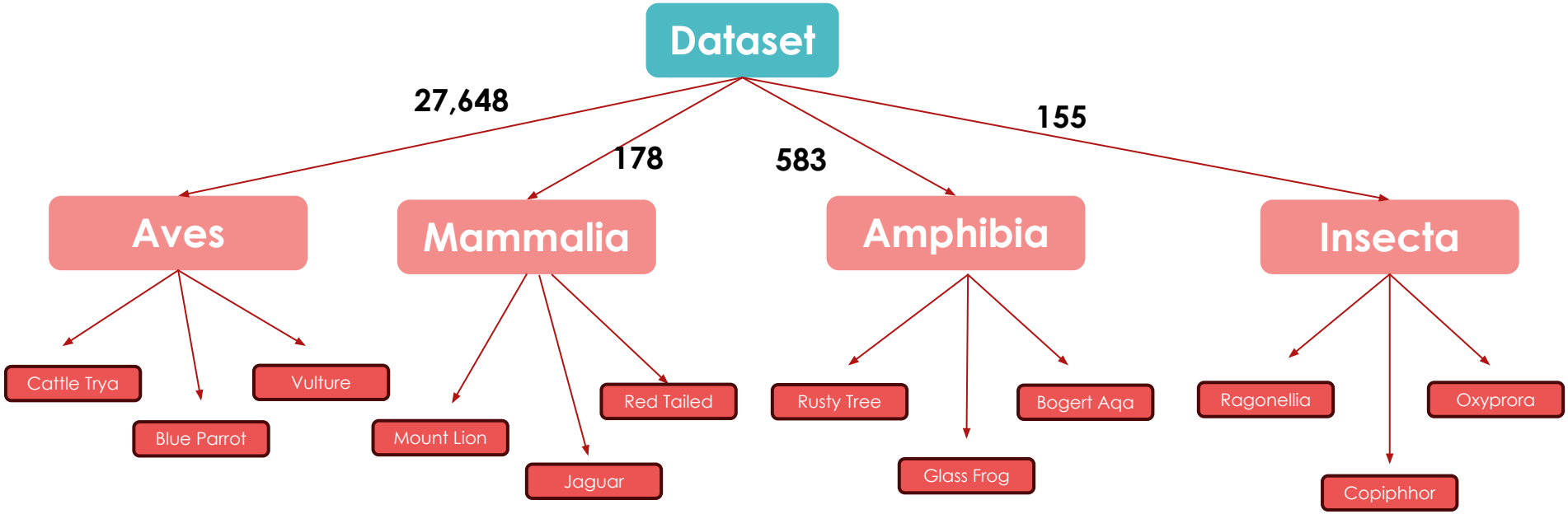
1. **Problem Statement**
2. **Data Description**
3. **From Audio to Images**
4. **Modeling Workflow**
5. **Results**
6. **Conclusion**

Problem Statement

- Given a set of audio files of bird sounds
- Train a deep learning model that learns to classify bird species based on sounds
- Deploy the trained model through a Streamlit app
- Make the app user friendly

Data Description

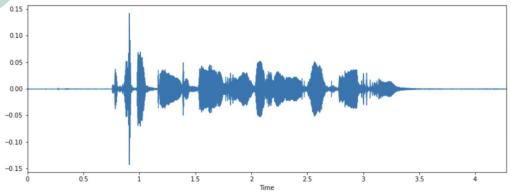
Number of audio files: 28,564
Main Classes: 4
Species Sub-Classes: 206



From Audio to Images

1

2

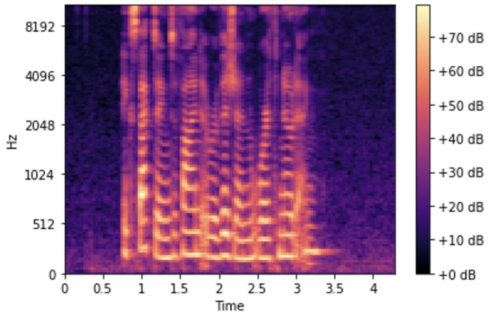


3

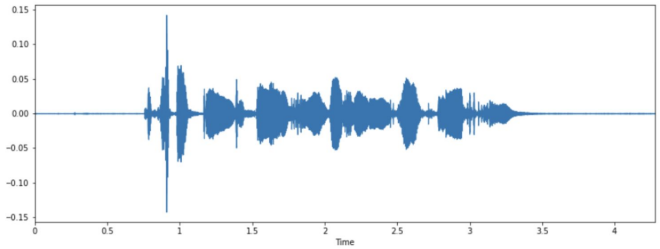
Fourier Transform

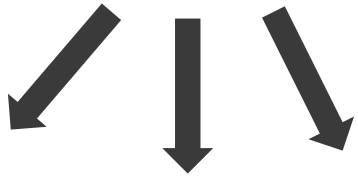
Library Used

4



Data Augmentation



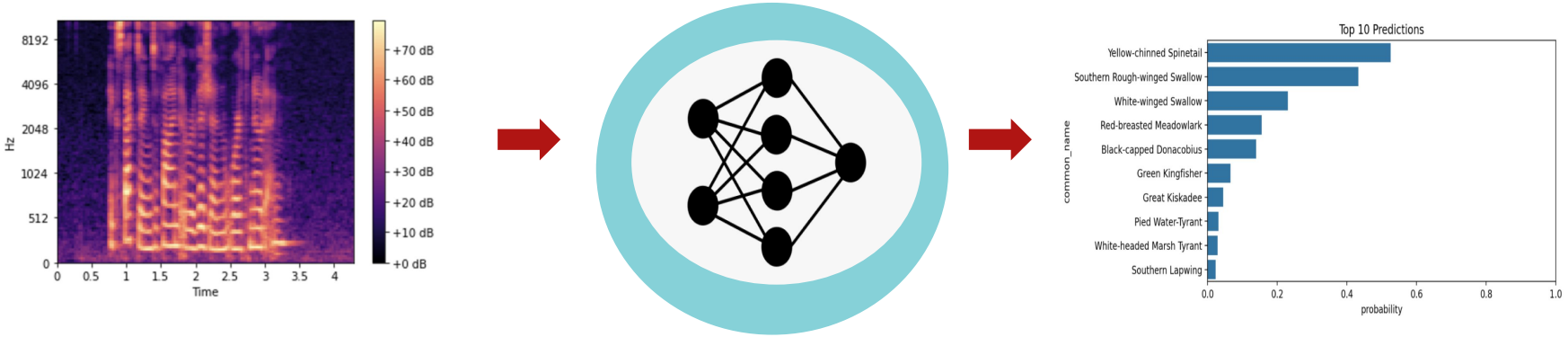


Time Stretching

Pitch Shifting

Volume Adjustment

Modeling

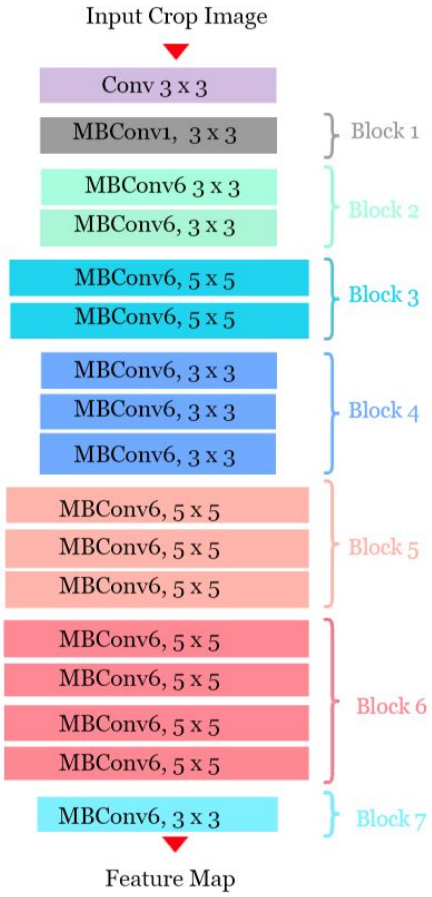


- MEL Spectrogram is converted into a tensor
- Tensor goes through Data Augmentation
- Model learns features from the image tensor and outputs a probability distribution across all classes

Project Structure

```
birdclef_project/
│
├── main.py                          # Entry point to run training/inference
├── config/
│   └── class_CFG.py                 # Configuration class
│
├── data/
│   └── class_BirdCLEFDatasetFromNPY.py  # Custom dataset class
│
├── models/
│   ├── class_BirdCLEFModel.py       # Model architecture
│   └── utils_model_definition.py    # Extra model components (layers, etc.)
│
├── train/
│   ├── util_run_training.py         # Main training orchestration
│   ├── utils_training_loop.py       # Training and validation loops
│   └── util_collate_fn.py           # Custom collate function for Dataloader
│
├── utils/
│   ├── utils_preprocessing.py       # Audio preprocessing, transformations
│   └── util_set_seed.py             # Seed-setting utility
│
├── test/                            # For future inference scripts / test code
│   └── (optional) inference.py      # Final model evaluation / prediction
│
└── README.md
```

EfficientNet-B0 Architecture

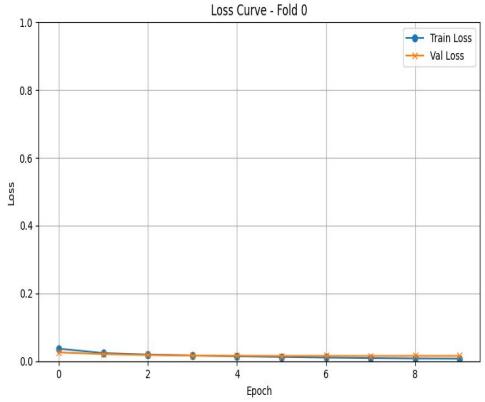


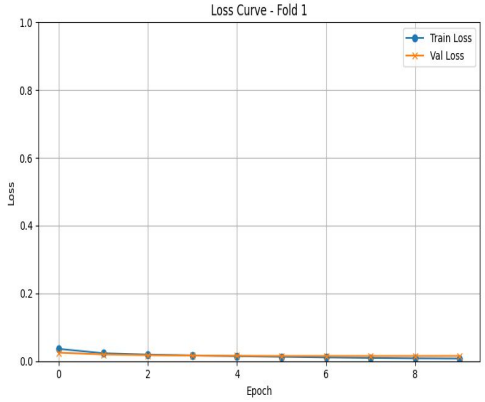
EfficientNet-B0 Baseline Network

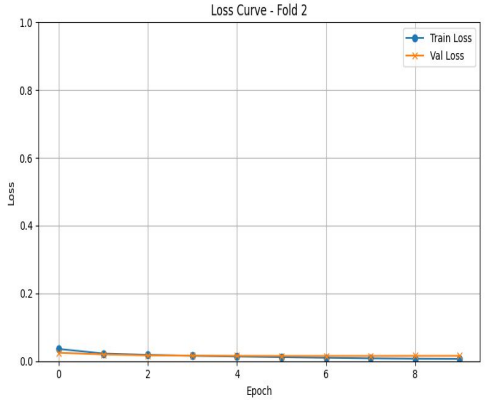
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

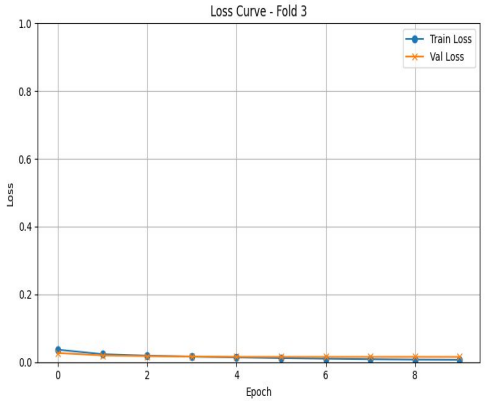
https://arxiv.org/abs/1905.11946

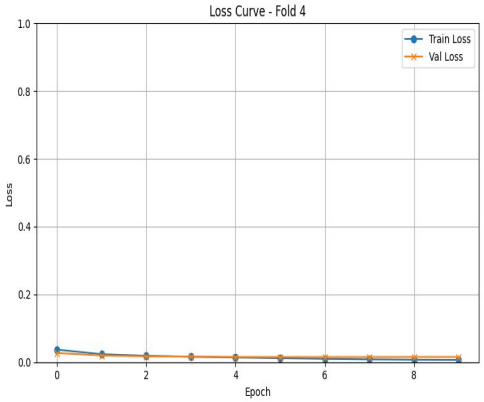
Model Evaluation - Loss



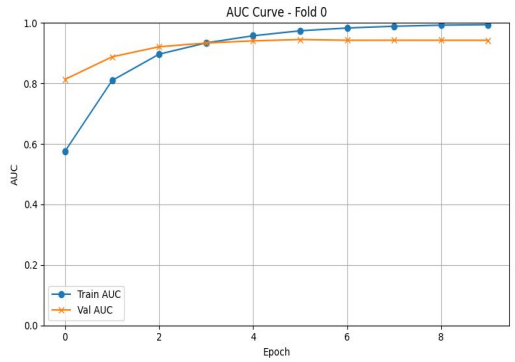


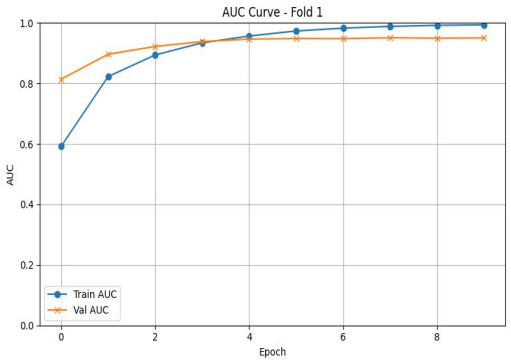


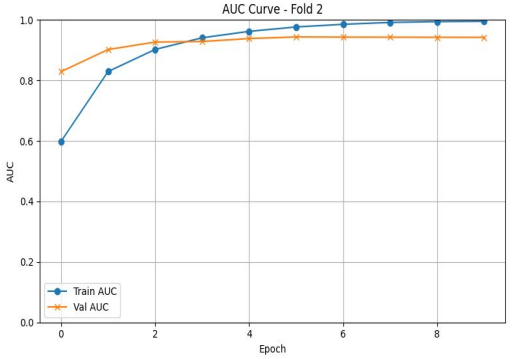


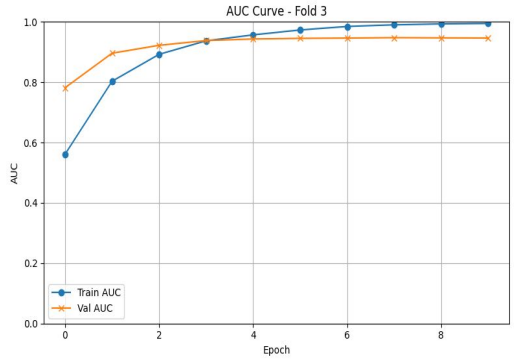


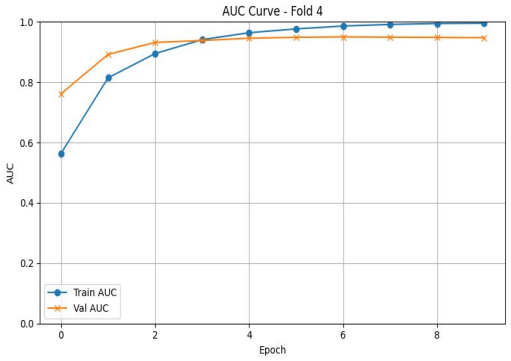
Model Evaluation - AUC







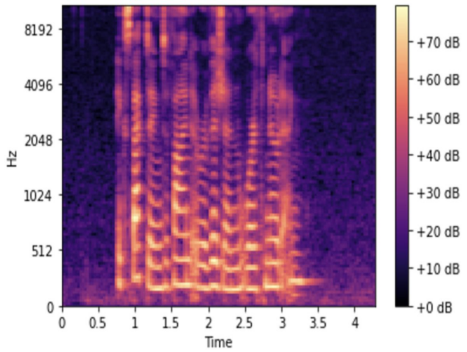





```
==================================
Cross-Validation Results:
Fold 0: 0.9451
Fold 1: 0.9513
Fold 2: 0.9438
Fold 3: 0.9475
Fold 4: 0.9501
Mean AUC: 0.9476
==================================
```

Modeling – Final Pipeline

MEL Spec Images | **Model Training** | **Save Model** | **Web App**



EffiecientnetB0 (CNN)

Trained Model

Conclusion

- Through this project we got to learn how audio can be used for image classification tasks
- We learned about the process of generating MEL Spectrograms
- Current model used is EfficientnetB0 but we can try it's deeper versions to see if that has any affect on the accuracy
- Overall the model performs decently well in terms of ROC AUC metric
- An interactive web app is also developed using Streamlit that loads the trained model and uses it for inference on a new audio sample