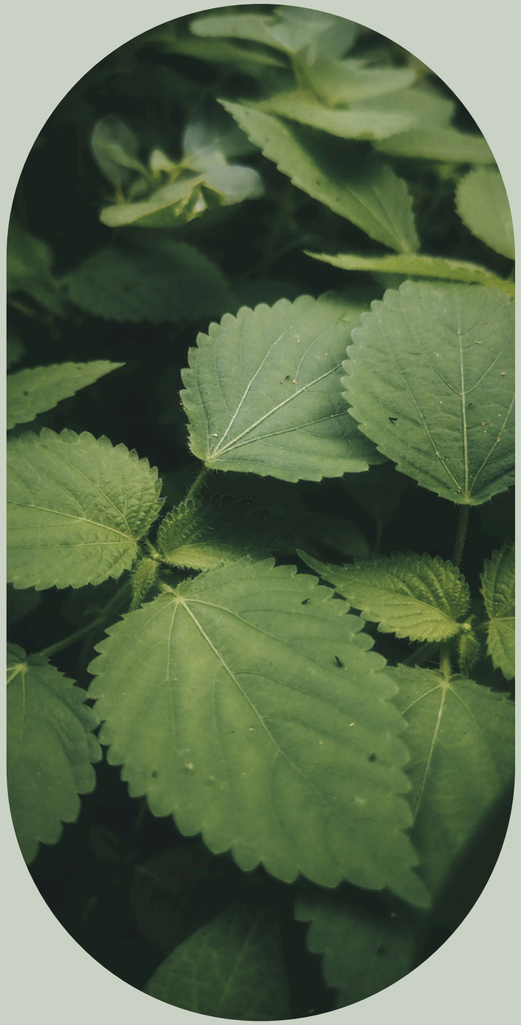
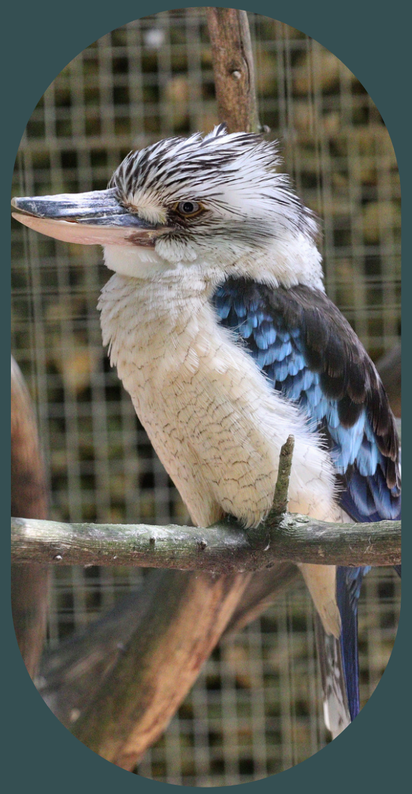
Final Project

# Birds Audio Classification

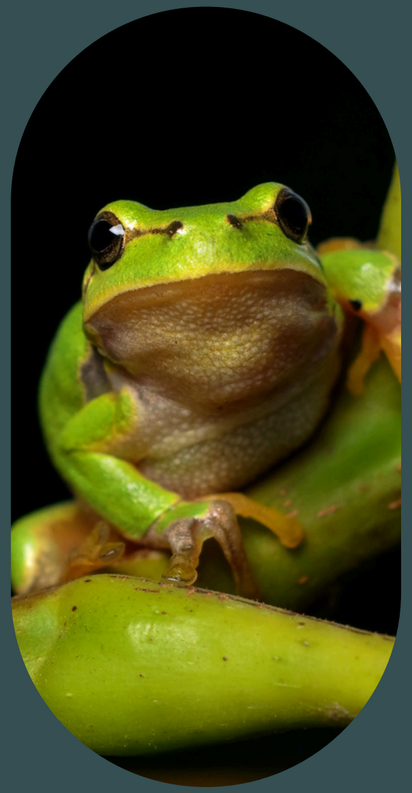


Leo Gordon, Andrew Fritz, Yen Nguyen, Nimay Goradia



# introduction

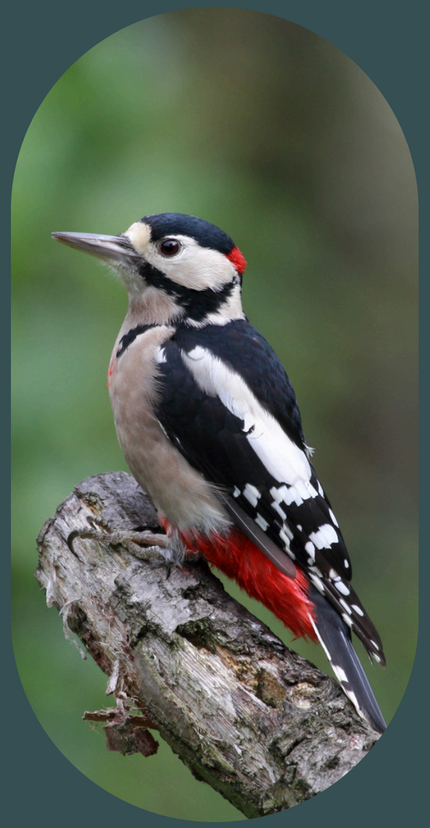
- Objective: Species identification from audio, focused on birds, amphibians, mammals and insects from the Middle Magdalena Valley of Colombia
- Given a set of labeled audio files, build a neural network YAMNet embeddings to accurately classify each sound.



# data

- BirdCLEF 2025 dataset from Kaggle - Audio files organized by class (species name)
- Audio files converted and preprocessed:
  - Mono conversion, downsampled to 22 kHz using *scipy.signal.resample*.
  - Cleaned up irregularities with a function *audio_data_unfucker*.

Final Project



# methodology

**Preprocessing:**

- Filtered small classes (manually sampled a subset).
- Ensured consistent audio format and data shape.
- Created a fixed-length audio representation.

**Feature Extraction:**

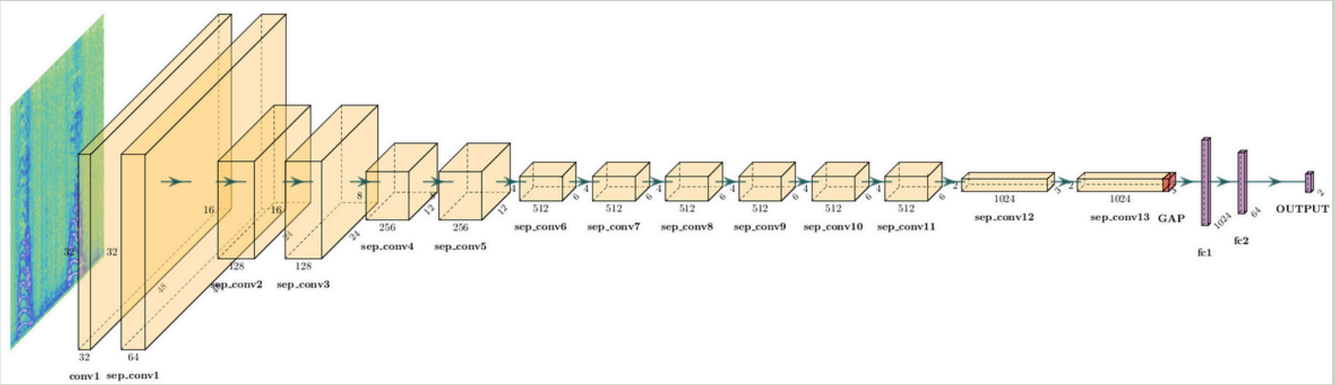
- Used YAMNet from TensorFlow Hub to convert audio into 1024-dimensional embeddings.

**Train-Test Split:**

- Used train_test_split from sklearn.
- Labels one-hot encoded with OneHotEncoder.

Final Project

# network architecture




YAMNet Architecture

- Input: 1024-dim YAMNet vector
- 3 Dense layers (200 units, ReLU)
- Dropout: 0.1 after each layer
- Output: Softmax (multi-class)
- EarlyStopping (patience = 3)

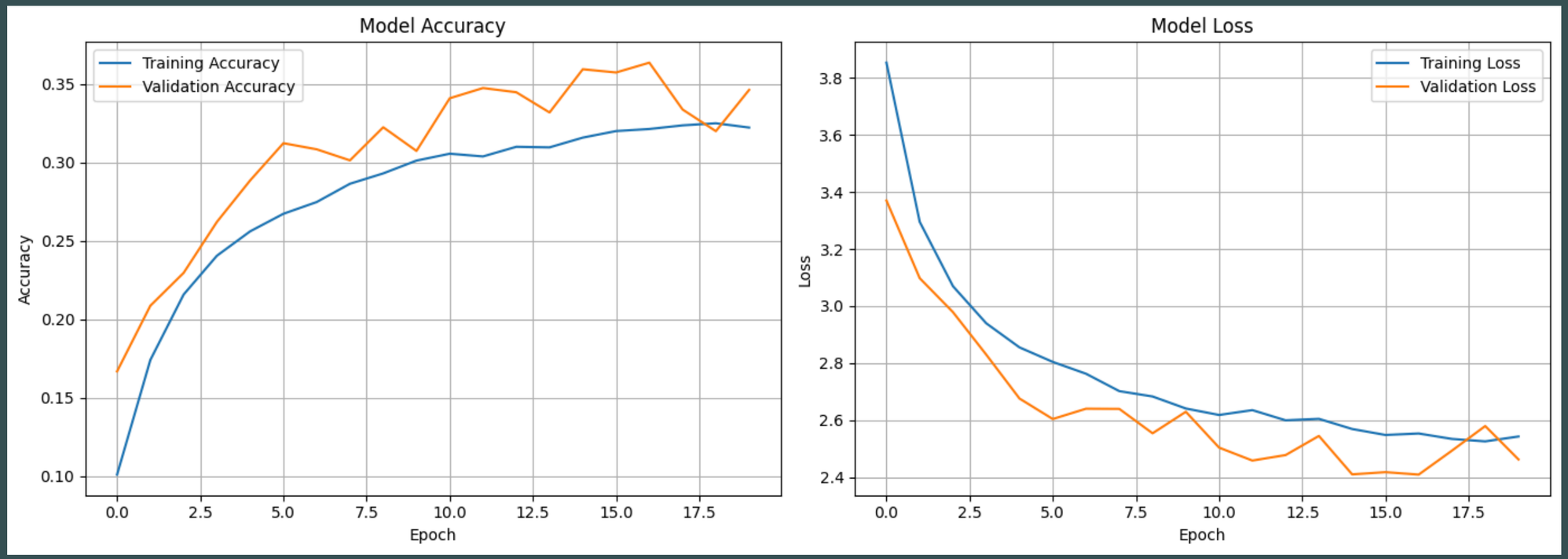
**Model: "sequential"**

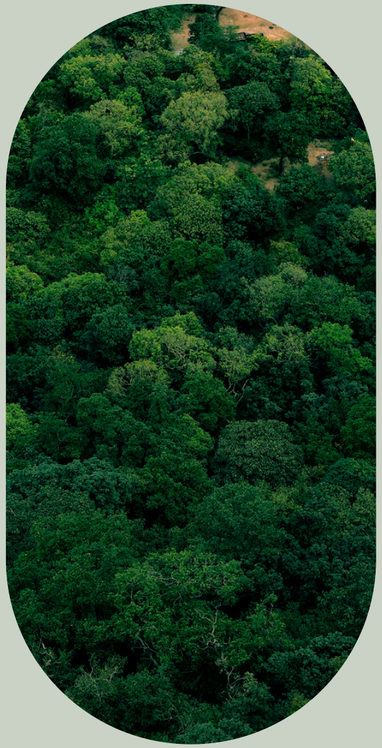
| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense (Dense) | (None, 200) | 205,000 |
| dropout (Dropout) | (None, 200) | 0 |
| dense_1 (Dense) | (None, 200) | 40,200 |
| dropout_1 (Dropout) | (None, 200) | 0 |
| dense_2 (Dense) | (None, 200) | 40,200 |
| dense_3 (Dense) | (None, 206) | 41,406 |

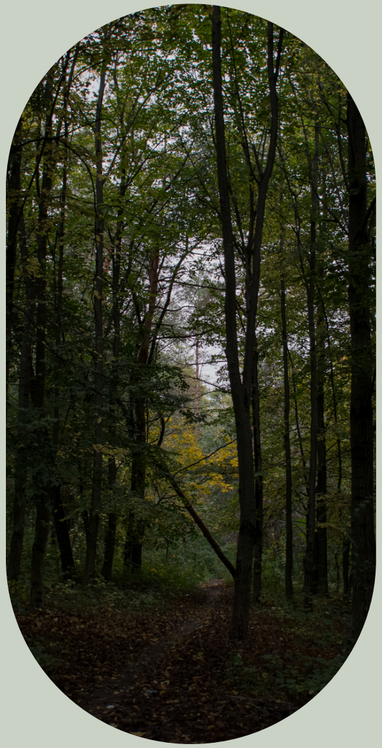
**Total params:** 326,806 (1.25 MB)
**Trainable params:** 326,806 (1.25 MB)
**Non-trainable params:** 0 (0.00 B)
Model compiled successfully.

# results

- Early Stopping at Epoch 17
- Test Accuracy: ~35%
- Test Loss: 2.41
- Signs of overfitting despite dropout







# take-aways

**Challenges:**

- Small sample per class led to overfitting.
- Imbalanced class distribution.

**What Helped:**

- Regularization via Dropout.
- Early stopping improved generalization.
- Manual class balancing (limited samples per class).

**Ideas for Future Work:**

- Train the model in batches to incorporate more of the dataset.
- Use a Kaggle cluster with more RAM or CPUs to load and process more data faster.
- Try more advanced models (e.g., LSTM, Transformer-based audio classifiers).