BIRD CALLS IN COLOMBIA: MACHINE LEARNING PRESENTATION

Presentation by Eric Steinberg, Veronica Vargas, Jacob Hillman, Eva Capelson

OBJECTIVE: *CLASSIFY BIRD SPECIES FROM AUDIO RECORDINGS USING ML.*

AUDIO ML IS CHALLENGING: OVERLAPPING CALLS, BACKGROUND NOISE, LONG AUDIO CLIPS.

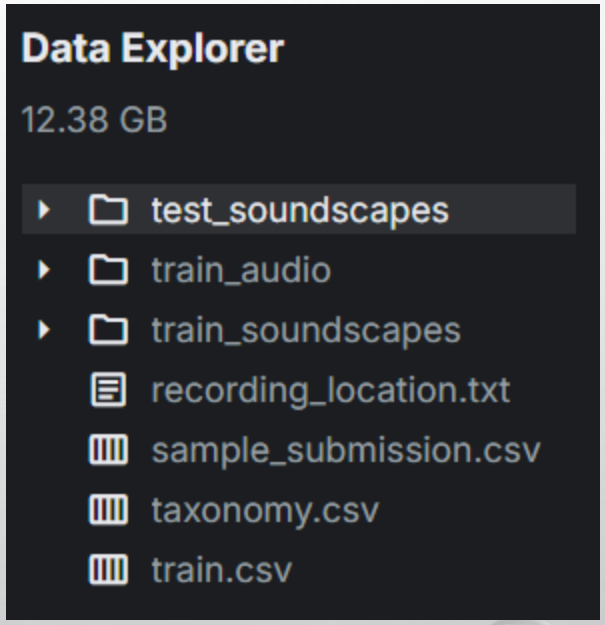
REAL-WORLD APPLICATIONS: CONSERVATION, BIODIVERSITY MONITORING.

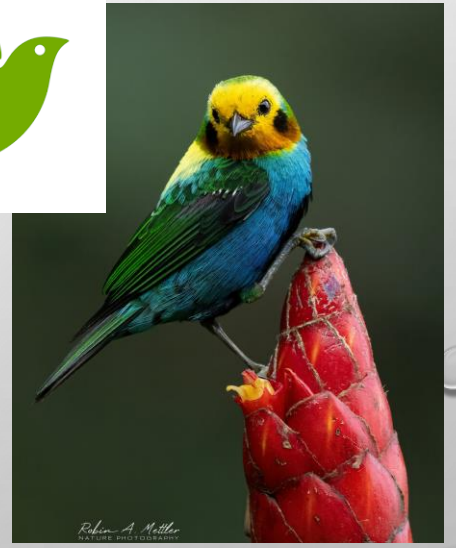
BIRDCLEFF-2025 DATASET

- DATA COLLECTED FROM RECORDINGS MADE IN EL SILENCIO NATURAL RESERVE, COLOMBIA
- 38.3K RAW AUDIO (.OGG) FILES
- 3 .CSV FILES
- 2 .TXT FILES

*STREAMLINE SPECIES CLASSIFICATION

*AID CONSERVATION RESEARCH





INTRODUCTION

Goal: Build an accurate classifier under computational (GPU/time) limits to compete with BirdNetLite (existing standard classifier)

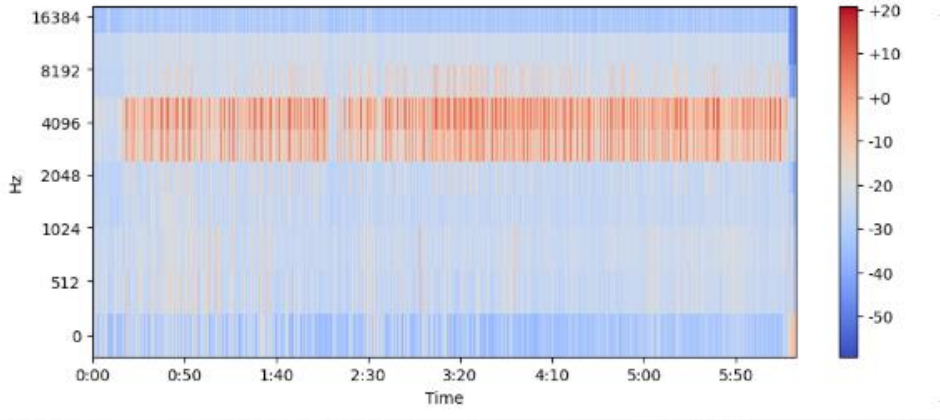
Key Components

- Audio Preprocessing: converting audio to spectrograms
- Models: Convolutional Neural Network
- Evaluation: precision, recall, F1-score

EARLY ATTEMPTS

- Learning from past Birdclef winners
- Simple preprocessing: short clips into spectrograms.
- Overly complex CNN models.
- Simple augmentation.
- **Result:** extremely low accuracy (1–2%), models struggled to learn anything.

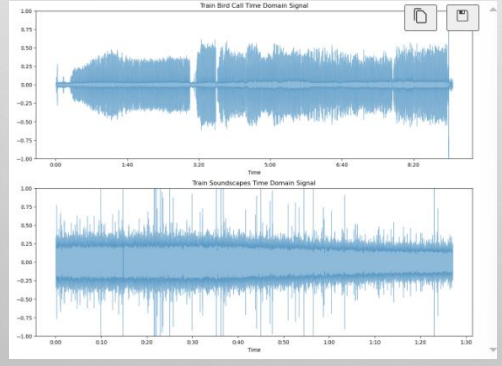
MODEL SELECTION

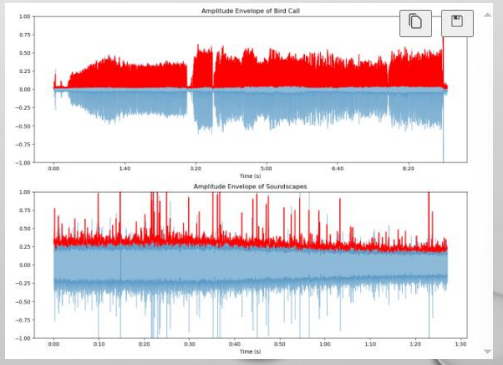


MACHINE LEARNING

- AMPLITUDE ENVELOPE (AE) **
- ROOT-MEAN SQUARE ENERGY (RMS) **
- ZERO CROSSING RATE (ZCR) **
- BAND ENERGY RATIO
- SPECTRAL CENTROID, ETC.

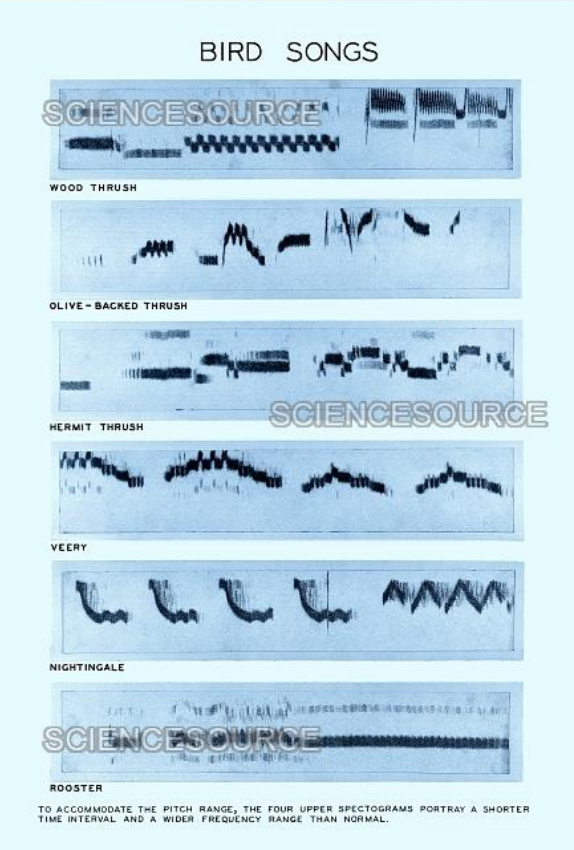
** = TIME DOMAIN AUDIO FEATURES





DEEP LEARNING

- PASS INDIVIDUAL FILE THROUGH MODEL
- CNN AND/OR RNN
- MEL-SPECTROGRAM
- AUTOMATIC FEATURE SELECTION



AUDIO PREPROCESSING PIPELINE

1. LOADING FILE
2. PADDING
3. EXTRACTING LOG SPECTROGRAM
4. SAVING SPECTROGRAM

THIS PIPELINE CONVERTS AUDIO FILES INTO .JPG FILES AND SAVES TO .NPY TO CONSERVE SPACE AND REDUCE COMPUTATIONAL INTENSITY.

TOOK SIGNIFICANTLY LESS TIME TO SAVE THAN ORIGINALLY SAVING FILES AS IMAGES.

WE SPLIT OUR DATA INTO 70% TRAINING AND 30% TESTING

Joining Pipeline

```
class PreprocessingPipeline:

    def __init__(self):
        self.padder = None
        self.extractor = None
        self.saver = None
        self._loader = None
        self._num_expected_samples = None

    @property
    def loader(self):
        return self._loader

    @loader.setter
    def loader(self, loader):
        self._loader = loader
        self._num_expected_samples = int(loader.sample_rate * loader.duration)

    def process(self, audio_files_dir):
        for root, _, files in os.walk(audio_files_dir):
            for file in files:
                file_path = os.path.join(root, file)
                self._process_file(file_path)
                print(f"Processed file {file_path}")

    def _process_file(self, file_path):
        signal = self.loader.load(file_path)
        if self._is_padding_necessary(signal):
            signal = self._apply_padding(signal)
        feature = self.extractor.extract(signal)
        self.saver.save_feature(feature, file_path)

    def _is_padding_necessary(self, signal):
        return len(signal) < self._num_expected_samples

    def _apply_padding(self, signal):
        num_missing_samples = self._num_expected_samples - len(signal)
        padded_signal = self.padder.right_pad(signal, num_missing_samples)
        return padded_signal
```

MODEL SETUP

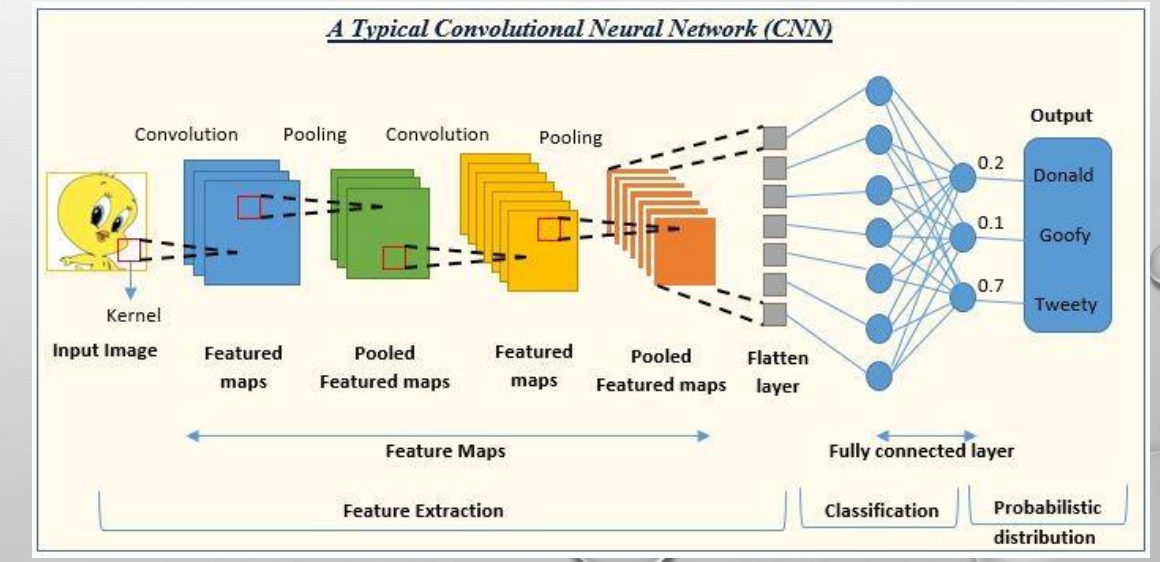
Problem Setup

- Network Structure:
  - Conv2D: Extract features from spectrograms.
  - MaxPool: Downsample feature maps.
  - Flatten: Convert to 1D for classification.
  - Fully Connected Layer: Outputs species predictions.
- Input: Spectrograms from audio data.
- Output: Class probabilities for bird species.

Experimental Setup

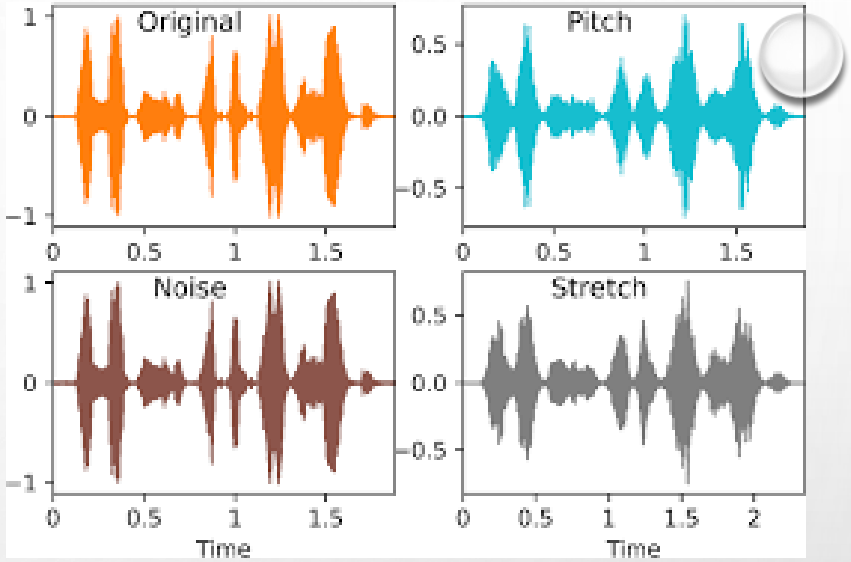
- Model: CNN (Conv2D, MaxPool, Flatten, Fully Connected Layer)
- Parameters:
  - Activation: ELU
  - Optimizer: Adam (learning rate = 0.001)
  - Batch Size: 32
  - Epochs: 50
- Environment: PyTorch, GPU-enabled machine.

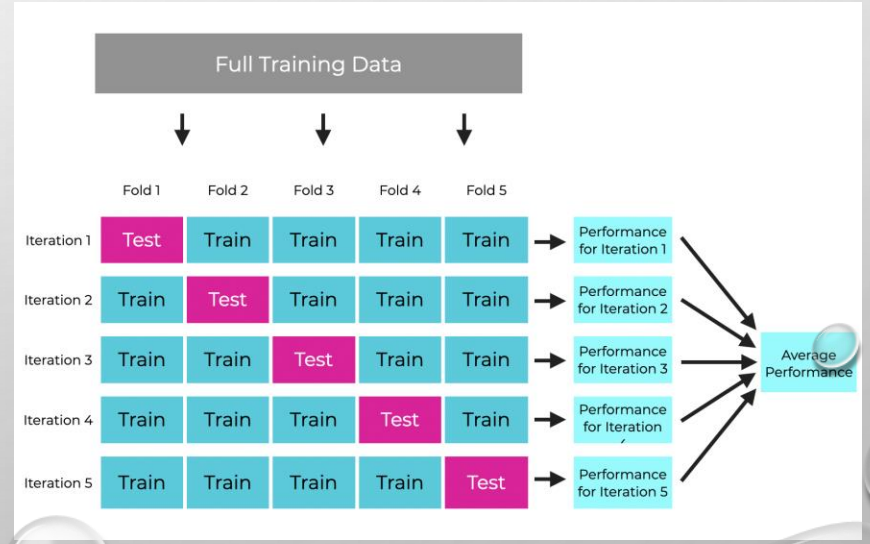
```
    num_classes: Any
) -> None
import t
class Ne
    def __init__(self, num_classes):
        super().__init__()
        # Define feature extractor
        self.feature_extractor = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),
            nn.ELU(),
            nn.MaxPool2d(kernel_size=2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ELU(),
            nn.MaxPool2d(kernel_size=2),
            nn.Flatten(),
        )
        # Define classifier
        self.classifier = nn.Linear(64 * 56 * 56, num_classes)

    def forward(self, x):
        # Pass input through feature extractor and classifier
        x = self.feature_extractor(x)
        x = self.classifier(x)
        return x
✓ 0.0s
```



COMPLEX FEATURES ADDED

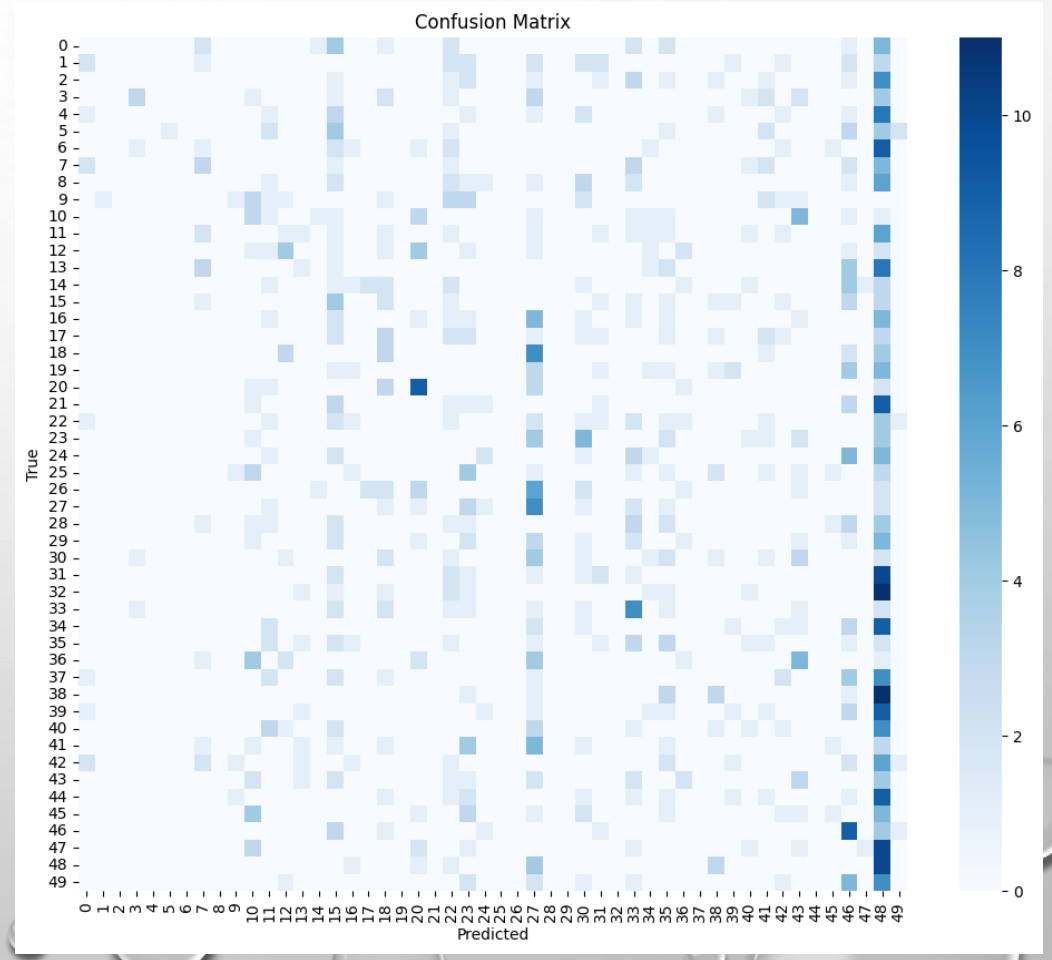
- **SUBSET SELECTION:** FOCUS ON **TOP 50 SPECIES** WITH MOST AUDIO ENTIRES
- GPU OPTIMIZED BATCH PROCESSING VIA CLOUD COMPUTING
- **STRONG AUGMENTATIONS (BOOTSTRAPPING):**
  - RANDOM TIME SHIFT (±0.5S),
  - RANDOM PITCH SHIFT (±1 SEMITONE),
  - GAUSSIAN NOISE INJECTION.
- **SPECTROGRAM CACHING:** SAVE .NPY FILES TO SKIP RECOMPUTATION. LOADED AND AUGMENTED ON THE FLY
- **CROSS VALIDATION:** 5-FOLD STRATIFIED KFOLD FOR FAIRER EVALUATION.





RESULTS (**COMPLEX** MODEL)

- Model able to recognize a few common species (e.g., Great Kiskadee — but only when overrepresented)
- Most species had **zero precision, recall, and F1** — model failed to predict them at all
- Severe **class confusion** — often predicted wrong species or nothing confidently
- **Macro average precision and recall ≈ 0** — model struggles especially with rare classes
- **Overall Accuracy ≈ 5%** — only slightly better than random chance among 50+ species
- Confusion matrix: very sparse, few correct predictions, mostly blank
- Indicates need for stronger models, more data, better preprocessing, or smarter label grouping



MAIN RESULTS (SIMPLE MODEL)

Precision(true +/total pred +): 0.24

Recall(pred +/all true +): 0.21

Cross Entropy Loss decreasing as we loop through training data to train the model

Epoch 1, Loss: 5.1880
Epoch 2, Loss: 4.7077
Epoch 3, Loss: 3.7155

	ID	Prediction
1	21038	0.0
2	21116	0.0
3	22333	1.0
4	24292	0.0
5	24322	0.0
6	41778	0.0
7	41970	0.0
8	42087	0.0
9	46010	0.0
10	476537	0.0
11	476538	0.0
12	48124	0.25

```python
import torch.optim as optim

# Define the model
net = Net(num_classes=52)
# Define the loss function
criterion = nn.CrossEntropyLoss()
# Define the optimizer
optimizer = optim.Adam(net.parameters(),lr=0.001)

for epoch in range(3):
    running_loss = 0.0
    # Iterate over training batches
    for images,labels in dataloader_train:
        optimizer.zero_grad()
        outputs = net(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    epoch_loss = running_loss / len(dataloader_train)
    print(f"Epoch {epoch+1}, Loss: {epoch_loss:.4f}")
```

LIMITATIONS (WE PICKED A PRETTY DIFFICULT PROJECT)

- Very small dataset for many species (few samples per class, hundreds of classes)
- Severe class imbalance (some species dominate)
- Bird calls often overlap with environmental noise, eachother
- Spectrograms lose fine-grained time information
- Limited compute resources restrict model size and tuning
- Clips trimmed for consistency (5 sec) might miss key events
- Basic augmentation not enough for full generalization
- A few weeks is not long enough to become an audio engineer

NEXT STEPS

- FOCUS ON ONLY 10 MOST COMMON BIRD SPECIES
  - **UPSAMPLE RARE SPECIES** OR **DOWNSAMPLE** OVER-REPRESENTED ONES TO BALANCE CLASSES.
- USE STRONGER AUDIO AUGMENTATIONS (SHIFT, PITCH, NOISE)
- UPGRADE MODEL TO MINI-RESNET FOR BETTER FEATURE EXTRACTION
- FINE-TUNE TRAINING SETTINGS (EARLY STOPPING, LEARNING RATE SCHEDULING)
- EXPLORE PRE-TRAINED CNNS (MOBILENETV2) IF NEEDED

- ON A POSITIVE NOTE: OUR FRAMEWORK IS VERY SIMILAR TO PAST SUCCESFUL BIRDCLEF WINNERS!
- ONCE MODEL IS ACCURATE, APPLY TO THE UNLABELED COLUMBIAN BIRD CALLS AND SUBMIT TO KAGGLE COMPETITION!

RESOURCES

Kaggle Competition Link: Https://Www.Kaggle.Com/Competitions/Birdclef-2025/Overview

Incze, A., Jancsó, H.-B., Szilágyi, Z., Farkas, A., & Sulyok, C. (2019). Bird Sound Recognition Using a Convolutional Neural Network. *ResearchGate*.

Gavali, P., Patil, N. C., & Mhetre, P. A. (2017). Bird Species Identification Using Deep Learning. *ResearchGate*.

Velardo, Valerio. "The Sound of AI." *YouTube*, YouTube, www.youtube.com/@ValerioVelardoTheSoundofAI. Accessed 26 Apr. 2025.