Bowel Sound Detection using Deep Learning

Presented by:

Fouzi BOUKHALFA, PhD - Eng

Presentation Plan: Bowel Sound Detection using Deep Learning

- Introduction & Motivation
- Dataset Overview
- System Architecture
- Stage 1: Classifier.
- Stage 2: Regression.
- Inference & Aggregation
- Results Summary
- Conclusion & Future Work

Introduction & Motivation

Goal: Develop a proof-of-concept ML model for identifying bowel sounds in audio data and differentiating between 3 main classes:

- Single burst (labelled b): "These are faint and comprise approximately 85% of all bowel sounds. They occur multiple times per second, typically last 10–40 milliseconds, and have a frequency range of 60 Hz to 2 kHz."[1]
- Multiple burst (labelled mb): "These represent clusters of single and distinct burst sounds occurring in quick succession. They account for roughly 5% of all bowel sounds and can last up to 1.5 seconds."[1]
- Harmonic (labelled h): "The rarest type, comprising about 1%, these sounds are irregular and can last for several seconds. They are often associated with audible stomach rumbling."[1]

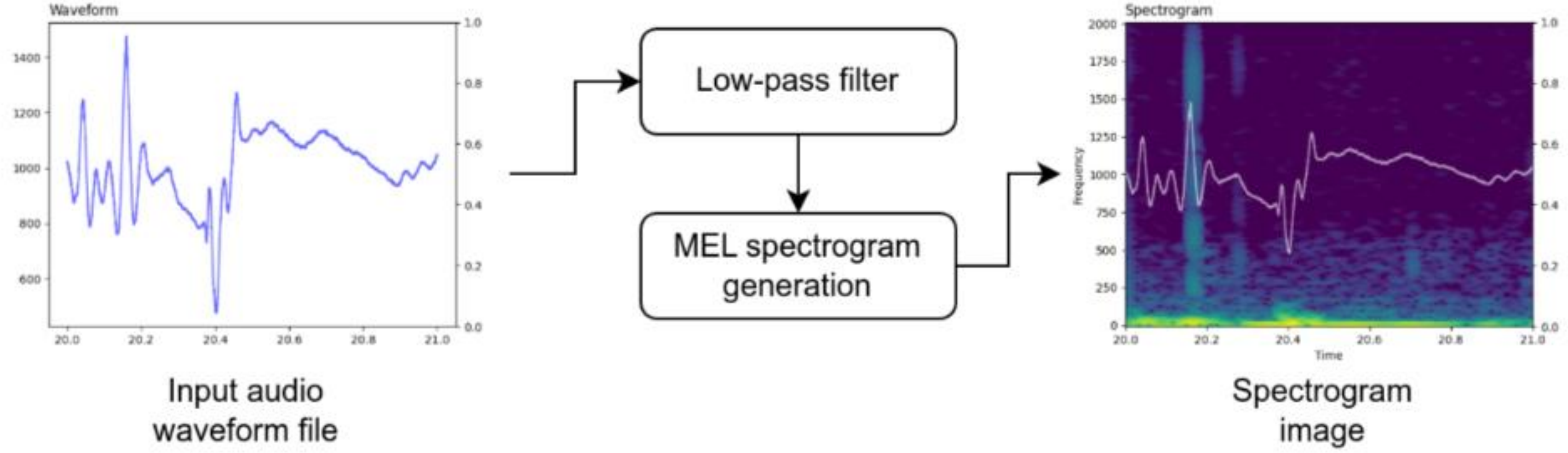
The model should identify the **start time**, **end time** and **type of each bowel sound.**

[1]. Matynia, I., & Nowak, R. (2025). BowelRCNN: Region-based Convolutional Neural Network System for Bowel Sound Auscultation. arXiv preprint arXiv:2504.08659.

Introduction & Motivation

How this task is solved in the literature (State of the Art):

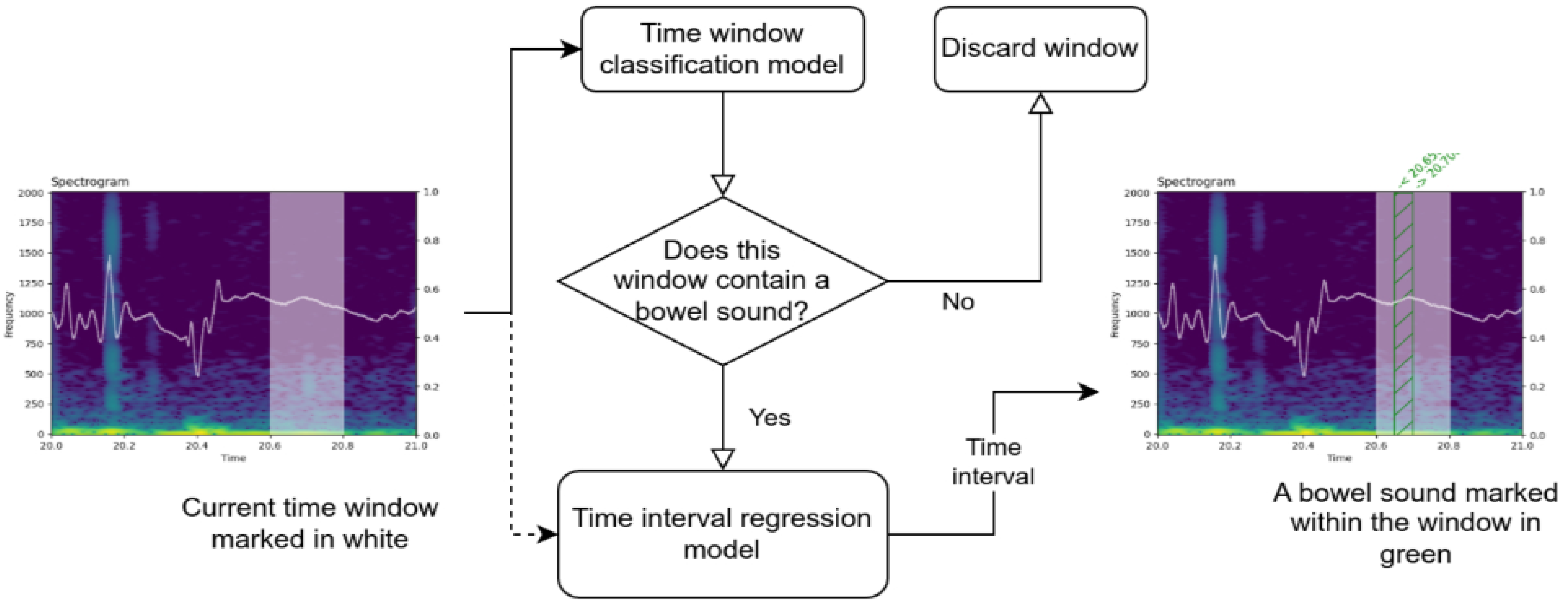
- **Frequency filtering**: Bowel sounds typically occur between **50 Hz and 2000 Hz**.
  → Apply a **low-pass filter** to isolate the **Signal of Interest (SOI)**.
- **Sliding time window**:
  The audio is **split into 0.2-second windows** using a sliding window approach.
- **Spectrogram conversion**:
  Each segment is transformed into a **time–frequency representation using Mel-spectrograms** (or other spectrogram types).



*SOI: Signal Of Interrest

[1]. Matynia, I., & Nowak, R. (2025). BowelRCNN: Region-based Convolutional Neural Network System for Bowel Sound Auscultation. arXiv preprint arXiv:2504.08659.

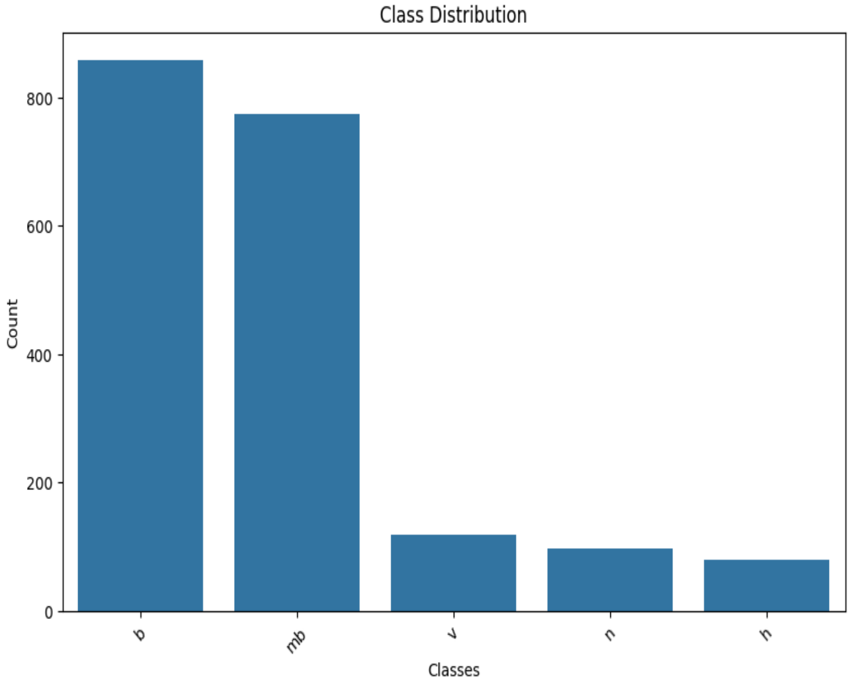
Introduction & Motivation



- Binary classification to detect the prescence of bowel sound by using CNN model.
- If yes, the regression model will predict whithin the slide window the offset (start time) and the scale factor.
- Aggregation of Predictions.

[1]. Matynia, I., & Nowak, R. (2025). BowelRCNN: Region-based Convolutional Neural Network System for Bowel Sound Auscultation. arXiv preprint arXiv:2504.08659.

Dataset Overview



	Start Time	End Time	Label
0	0.000000	17.019255	n
1	0.988978	1.242612	h
2	1.278083	1.447401	b
3	1.496558	1.644029	b
4	1.731420	1.851581	b
...	...	...	...
95	88.176780	88.362485	mb
96	88.384332	88.509956	b
97	88.559113	88.701122	b
98	88.842846	91.219054	h
99	89.274620	89.394782	b

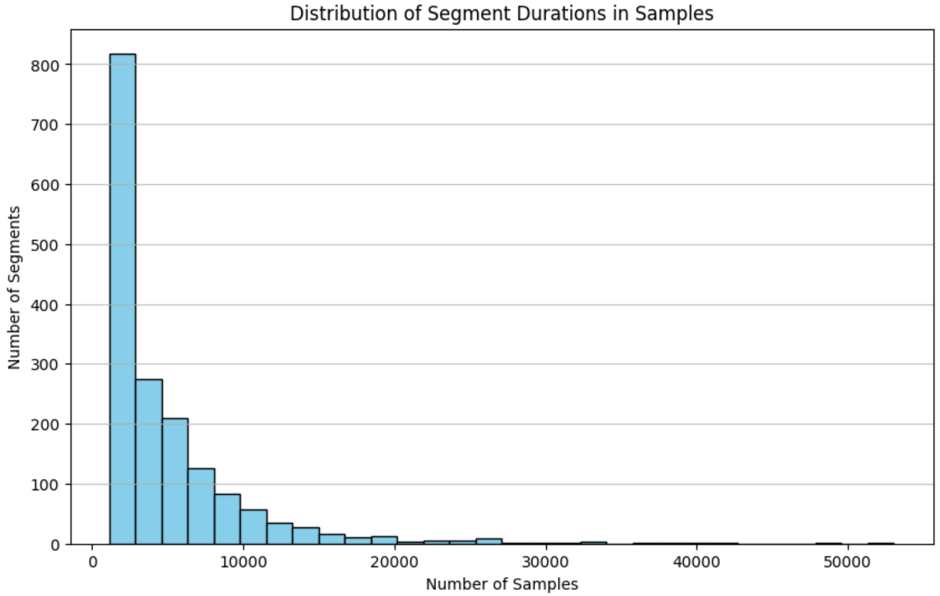
100 rows × 3 columns

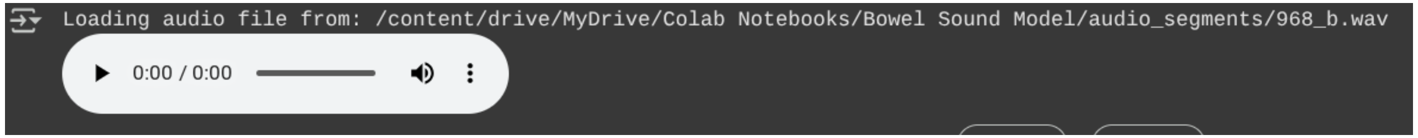
To resolve Unbalance classes:
=> Using weighted cross-entropy
=> Data augmentation.
=> Focal loss.

Label	
b	857
mb	773
h	80

Dataset Overview

Duration imbalance :





Listen to each category ("b","mb","h") with Ipython

Dataset Overview

📦 Data Preparation:

- AS_1 : was split into tarin_set and validation set respectively with the following percentage: **80%, 20%**.
- 23M24M: was used for inference as unseen data test_set.

System Architecture

- **Sound Type Classification**:
The type of each bowel sound is a **discrete label** (e.g., b, mb, h, none).
→ This is handled using a **classification model**

- **Start Time & End Time Prediction:**
These are continuous values, so we formulate this as a regression problem.

✓ **Structure:** modular design with 2 stages.

1. **Stage 1** – Classifier: Is there a bowel sound? Which type?
2. **Stage 2** – Regressor: When (start, end)?
3. **Post-processing**: Merge overlapping or closely spaced predictions of the same class into a single, aggregated detection.

All unlabeled audio segments are automatically treated as **background** and labeled `"no bowel sound"`.

Stage 2 is activated only if stage 1 detect a bowel sound with high confidence

[Audio] → [Segment → Spectrogram] → [Classifier: "no", "b", "mb", "h"]

↓

If class ≠ "no bowel sound"

→ Regressor (start, end)

System Architecture

🔧 Preprocessing Pipeline: From Raw Audio to Model Input:

🎧 1. Audio Loading

- Load .wav file using Librosa with original sampling rate.

🧽 2. Low-Pass Filtering

- Apply a Butterworth filter (cutoff = 2KHz)
  → Removes high-frequency noise, keeps bowel-relevant frequencies.

✂️ 3. Segmentation

- Divide audio into overlapping segments:
- Segment duration: 0.2s
- Stride: 0.05s

🔁 4. Data Augmentation

- Add Gaussian noise (std = 0.005) during training
  → Improves generalization, combats overfitting.

📊 **5. Mel-Spectrogram Conversion**

- Compute **128-band Mel-spectrograms**
- n_fft = 512, hop_length = 128
  → Converts time domain → frequency domain

📉 **6. Log Scaling**

- Convert to **dB scale** using power_to_db()
  → Compresses dynamic range.

🧩 **7. Spectrogram Normalization**

- **Pad** all spectrograms to fixed size: **(128 × 32)**
  → Ensures input consistency for the CNN.

🏷️ **8. Label Assignment + Encoding**

- For each segment:

Match against annotation intervals to assign class (b, mb, h, or none)

- Normalize **start/end times** for regression
- **Encode class labels** as integers using:

{'b': 0, 'mb': 1, 'h': 2, 'none': 3}

System Architecture

🛡️ Detect overfitting by plotting the train and validation loss.

🛡️ Preventing Overfitting in Bowel Sound Detection:

✅ **Data Augmentation**

- **Gaussian Noise Injection** on audio segments during training (augment=True)
  → Simulates real-world variability in bowel sounds
  → Improves model generalization

✅ **Regularization Techniques**

- **Dropout Layers** in both Stage 1 and Stage 2 models (30% in Stage 1, 20% in Stage 2)
- Prevents co-adaptation of features

- **Weight Decay (L2 regularization)** in optimizers

Penalizes large weights in the network
Helps control model complexity

✅ **Label Smoothing via Class Weighting**

- **Weighted Cross-Entropy Loss** accounts for class imbalance

Reduces model bias toward frequent classes
Prevents overfitting to dominant labels (e.g., 'b', 'mb')

Stage 1: Bowel Sound Classification (Detection Model)

Objective:

Classify each audio segment into one of 4 categories:`{b: burst, mb: multiple bursts, h: harmonic, none}`

Model Architecture:

Input: 1-channel mel-spectrogram (`[1, 128, 32]`)

- **Conv Layer 1**: 32 filters, 3×3 kernel + ReLU + AvgPool (2×2)
- **Conv Layer 2**: 64 filters, 3×3 kernel + ReLU + AvgPool (2×2)
- **Conv Layer 3**: 128 filters, 3×3 kernel + ReLU + AvgPool (2×2)
- **Dropout**: 30% (to reduce overfitting)

Fully Connected Layers:

FC1: 128 neurons + ReLU

FC2: 4 output neurons (softmax for class prediction)

Loss Function:

- Weighted CrossEntropy Loss to handle class imbalance (rare events like h).

Training Strategy:

- Stratified data split into train/validation sets.
- Includes data augmentation with Gaussian noise.
- Evaluated using accuracy, loss, and confusion matrix.

Output:

- Class probabilities over 4 bowel sound types

Stage 2 – Regression Model: Precise Interval Localization

Purpose: Predict the **start** and **end** (normalized) of the detected bowel sound segment within the audio.

Input:
Same mel-spectrogram as Stage 1(b, mb, h).

Output:
Start: Predicted relative start time (within the segment)
End: Predicted relative end time

Model Architecture:
- **Conv Layer 1**: 32 filters + ReLU + AvgPool (2×2)
- **Conv Layer 2**: 64 filters + ReLU + AvgPool (2×2)
- **Conv Layer 3**: 128 filters + ReLU + AvgPool (2×2)
- **Dropout**: 20% to prevent overfitting
- **Fully Connected Layer**:

FC1: 128 neurons + ReLU
- **Two Heads** (Regression Outputs):

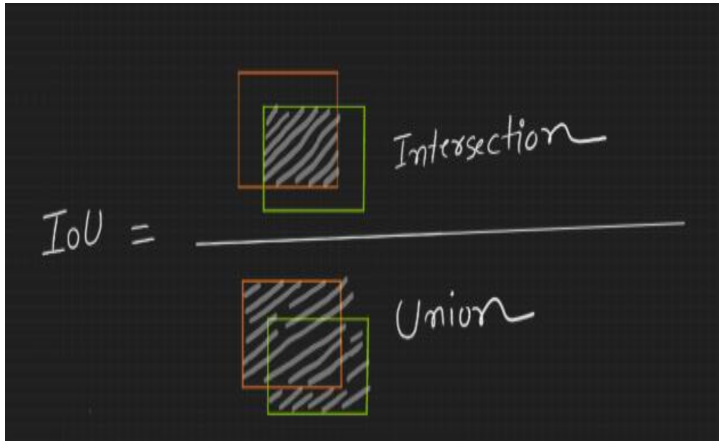
fc_scale: predicts **duration** (end - start)
fc_offset: predicts **start time**
Both use **sigmoid** to constrain values in [0, 1]

Loss Function:
Custom **IoU-based loss** that penalizes:
✔ Poor overlap (IoU)
✔ Misaligned midpoints
✔ Incorrect duration (scale mismatch)

Training:
- Train the model only on positive segment.
- Supervised regression using annotated event intervals.

Advantage:
Adds **temporal precision**, enabling accurate localization

Stage 2 – Regression Model: Precise Interval Localization



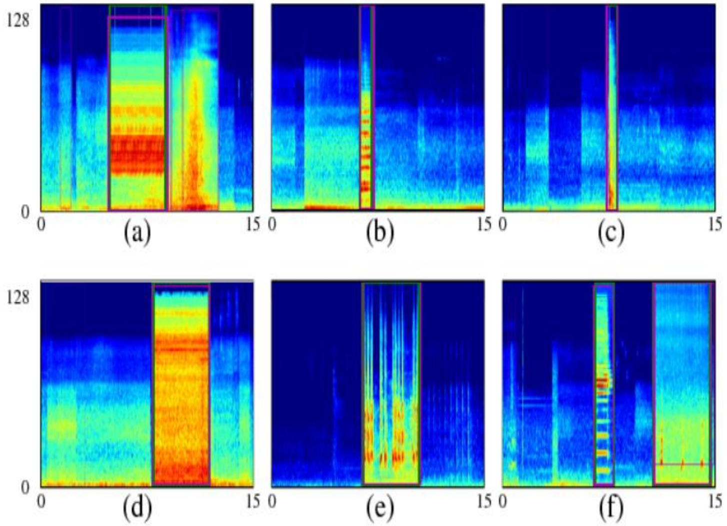
Custom IoU-Based Loss – Why?

Use Case:

- Bowel sound **segment localization** in time series
- Predict **start** and **end** times of sound events

Why This Loss?

- ✅ **IoU term**: Measures temporal overlap accuracy
- ✅ **Mid-point penalty**: Aligns predicted center with true center
- ✅ **Scale penalty**: Matches predicted duration with true duration
- ✅ Robust to small errors, focuses on **interval quality**, not just point estimates



Why Not MSE?

- Ignores overlap and structure of intervals
- Sensitive to outliers, doesn't penalize overlap errors well.

Summary:

- Captures event **position + duration + overlap**
- Better suited for **temporal event detection** than naive regression

Component	Penalizes When...	Encourages
`iou_loss`	No or small overlap	High overlap
`dist_penalty`	Center is misaligned	Center alignment
`scale_penalty`	Predicted duration is incorrect	Matching duration

Inference & Aggregation

Merging Overlapping Detections

Because the model analyzes overlapping segments, it may detect the same bowel sound multiple times across neighboring windows. To avoid duplicate detections:

- Predicted intervals are grouped if they are **close in time** and **predicted to be the same class**.
- The overlapping predictions are merged into a **single event** using a **voting** and **confidence aggregation scheme**.

✔️ This step improves the temporal precision of the detections and removes redundancy.

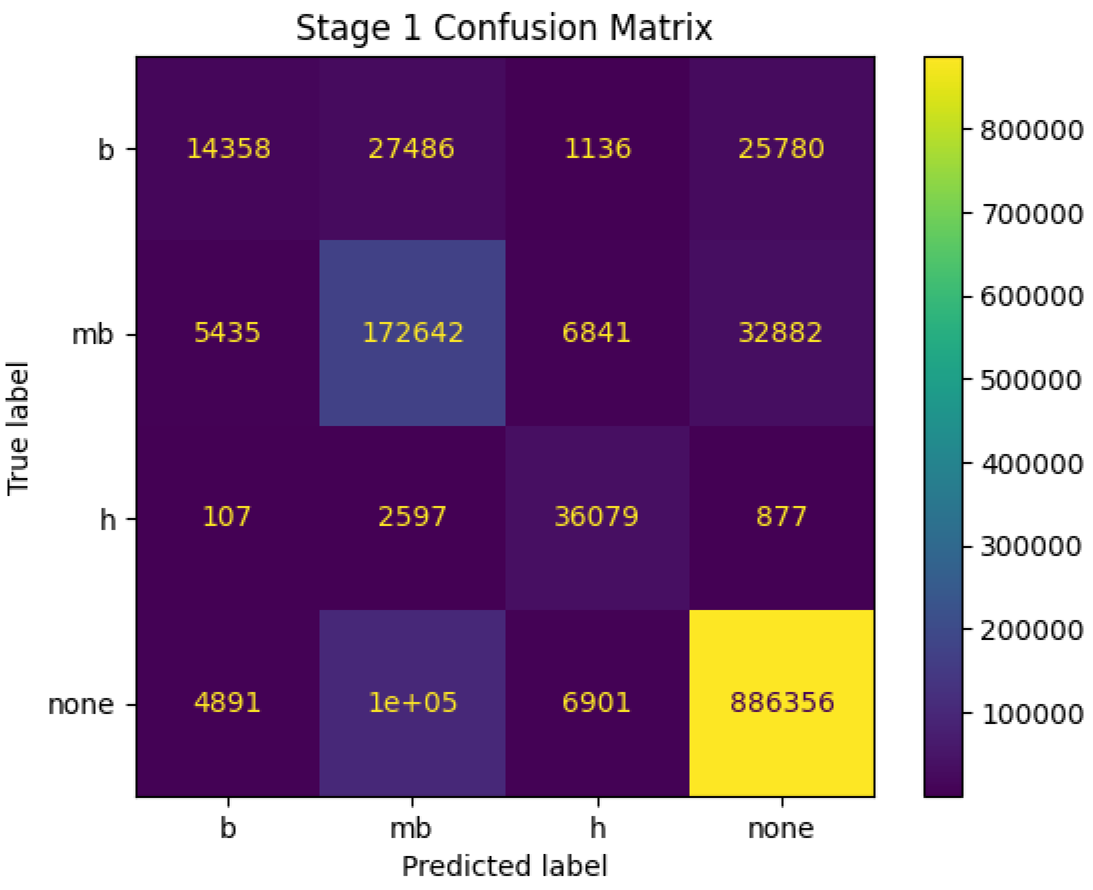
🎯 Filtering using Thresholds & Confidence Scores

Not all detections are reliable, so we filter out:

- **Low-confidence predictions** (e.g., classifier probability < 0.9).
- **Merged groups** whose total combined confidence is **below a vote threshold** (e.g., < 1.5).

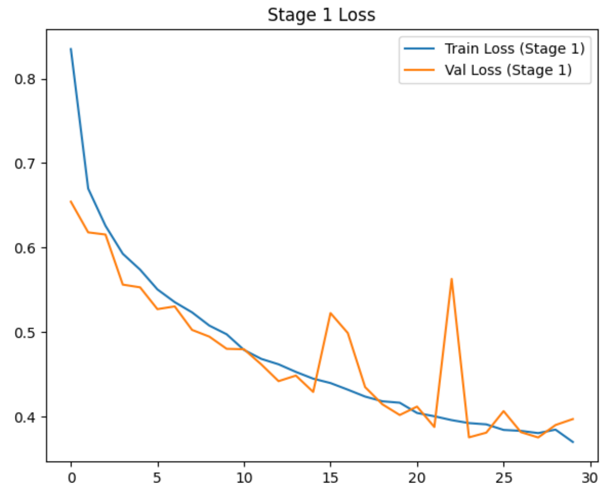
✔️ This ensures that only **high-confidence**, consistent detections make it to the final output.

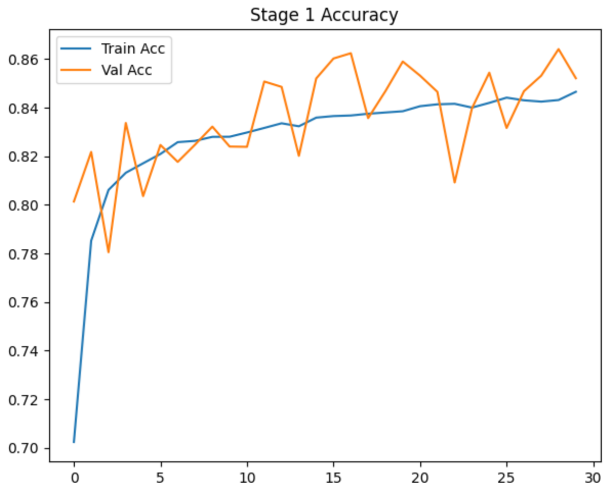
Results Summary: Classification



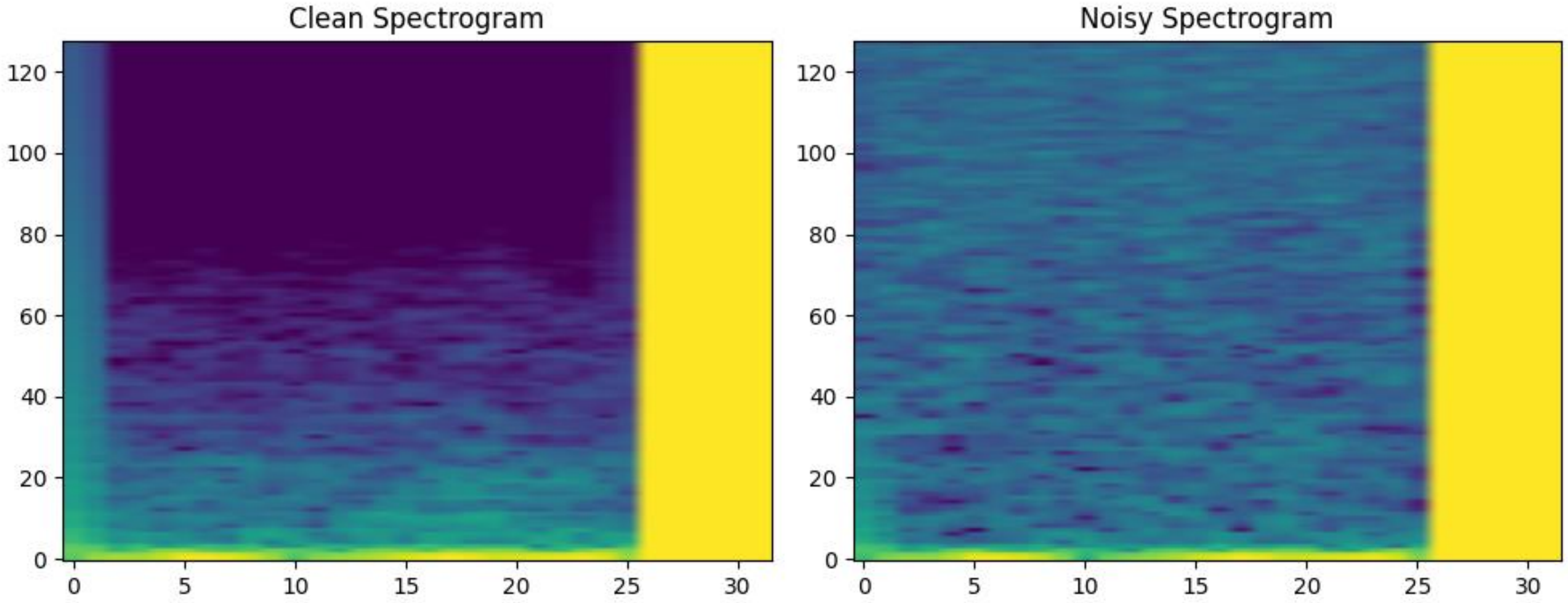
Classification Report (Stage 1):

	precision	recall	f1-score	support
b	0.58	0.21	0.31	68760
mb	0.56	0.79	0.66	217800
h	0.71	0.91	0.80	39660
none	0.94	0.89	0.91	1001130
accuracy			0.84	1327350
macro avg	0.70	0.70	0.67	1327350
weighted avg	0.85	0.84	0.83	1327350

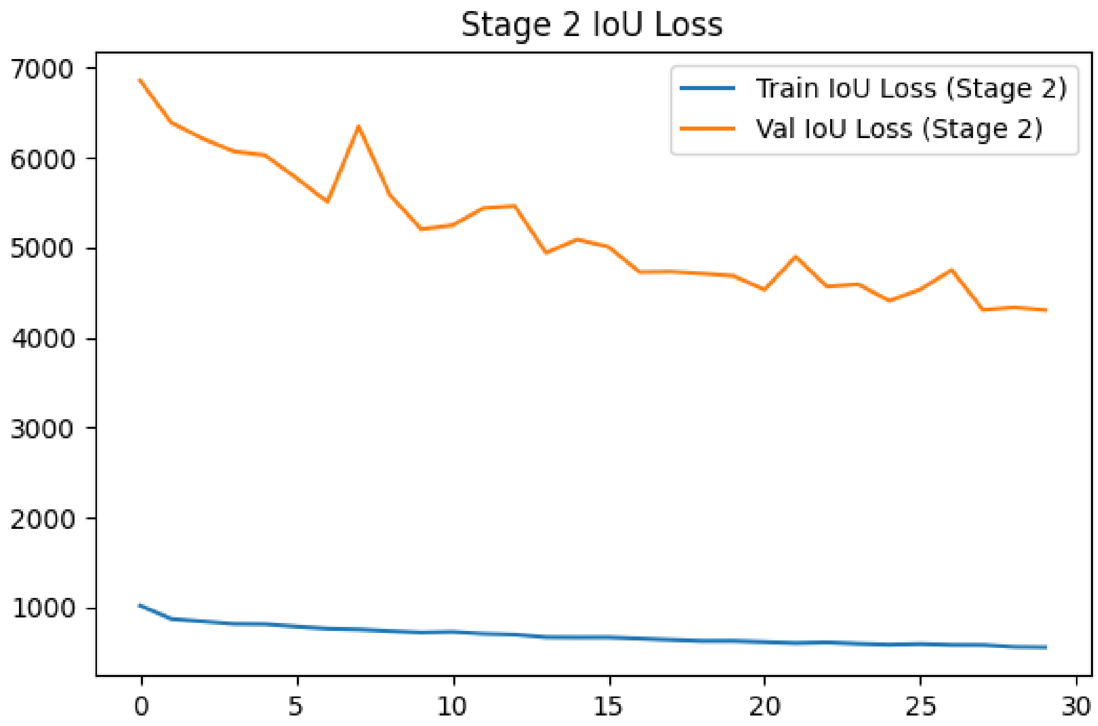


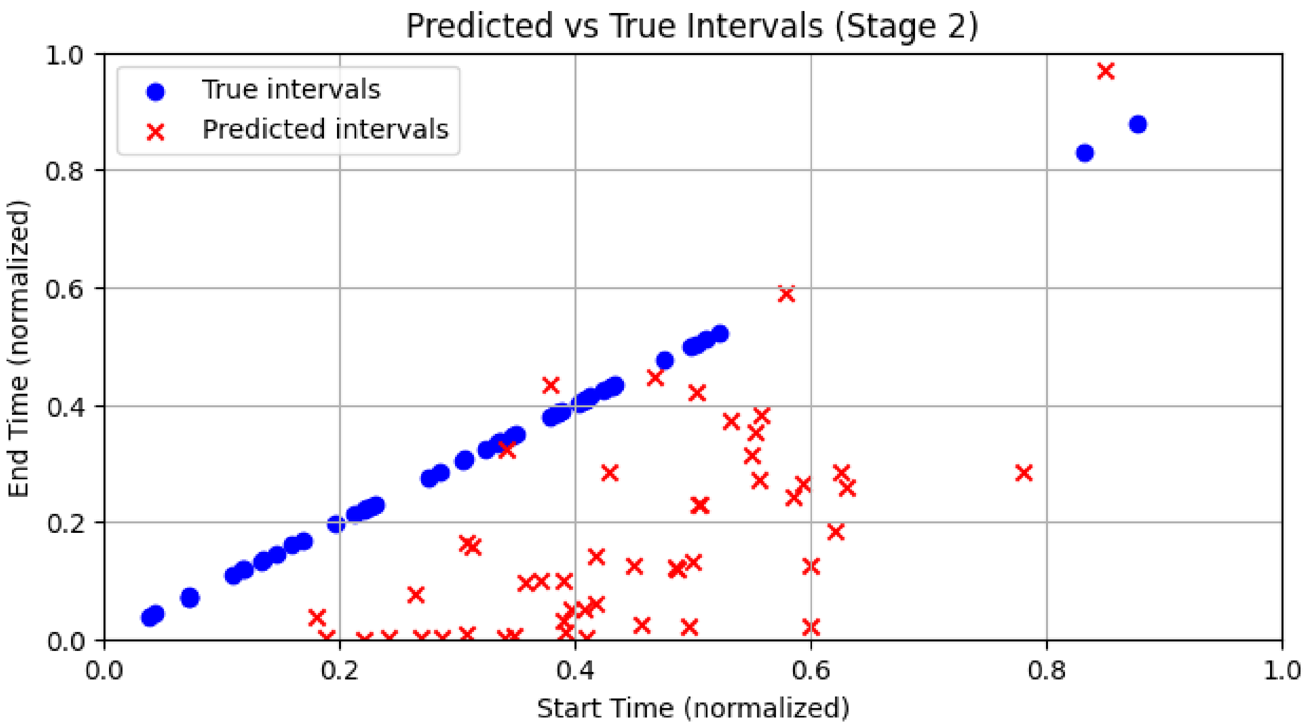


Results Summary: Data augmentation (Gaussien noise with std=0.005)



Results Summary: Regression





Lr= 0.0001 And Dropout 0.2

Insufficient epochs: The model still be converging! Which explains why the red dot is not fitting well at this stage.

Results Summary: Inference & Aggregation

Example:

- Sorts all raw detections by time.
- Merges overlapping predictions that:

Are close together in time (merge_gap ≤ 0.1s),

Are of the same class.

- Combines start and end times from the group.
- Sums confidence scores.
- Keeps the merged detection **only if** the total confidence exceeds vote_threshold (e.g., 1.5).

Start Time: 50.86s, End Time: 52.98s, Type: h, Confidence: 35.8
Start Time: 109.47s, End Time: 111.72s, Type: mb, Confidence: 22.47

These indicate the system is not only detecting, but aggregating (overlaped time intervals) and merging detections effectively.

```
Start Time: 1.03s, End Time: 1.93s, Type: h, Confidence: 3.9
Start Time: 13.2s, End Time: 13.83s, Type: mb, Confidence: 1.89
Start Time: 46.2s, End Time: 46.65s, Type: mb, Confidence: 3.85
Start Time: 47.39s, End Time: 48.61s, Type: mb, Confidence: 7.77
Start Time: 49.48s, End Time: 50.28s, Type: mb, Confidence: 3.87
Start Time: 50.86s, End Time: 52.98s, Type: h, Confidence: 35.8
Start Time: 58.75s, End Time: 59.48s, Type: h, Confidence: 8.91
Start Time: 60.4s, End Time: 60.56s, Type: mb, Confidence: 1.96
Start Time: 66.4s, End Time: 66.72s, Type: mb, Confidence: 1.9
Start Time: 67.6s, End Time: 67.8s, Type: b, Confidence: 1.99
Start Time: 69.5s, End Time: 69.69s, Type: mb, Confidence: 2.82
Start Time: 71.05s, End Time: 71.79s, Type: mb, Confidence: 2.89
Start Time: 72.7s, End Time: 73.52s, Type: h, Confidence: 7.83
Start Time: 77.3s, End Time: 77.52s, Type: b, Confidence: 1.97
Start Time: 78.25s, End Time: 78.77s, Type: mb, Confidence: 3.84
Start Time: 84.54s, End Time: 85.16s, Type: mb, Confidence: 1.88
Start Time: 87.3s, End Time: 87.38s, Type: mb, Confidence: 1.92
Start Time: 88.84s, End Time: 89.71s, Type: h, Confidence: 13.92
Start Time: 89.5s, End Time: 91.45s, Type: h, Confidence: 33.88
Start Time: 91.35s, End Time: 92.75s, Type: mb, Confidence: 21.45
Start Time: 92.55s, End Time: 93.07s, Type: h, Confidence: 9.0
```

Grid Search For Hyper-Parameters Tuning

- **Objective:**
  Optimize model performance by systematically searching combinations of hyper-parameters.

Stage 1

```
🔍 Trying combination: {'lr': 0.0001, 'batch_size': 16, 'weight_decay': 1e-05, 'dropout': 0.3}
Val Accuracy: 0.8025

🔍 Trying combination: {'lr': 0.0001, 'batch_size': 32, 'weight_decay': 0.0001, 'dropout': 0.2}
Val Accuracy: 0.8044

🔍 Trying combination: {'lr': 0.0001, 'batch_size': 32, 'weight_decay': 0.0001, 'dropout': 0.3}
Val Accuracy: 0.8030

🔍 Trying combination: {'lr': 0.0001, 'batch_size': 32, 'weight_decay': 1e-05, 'dropout': 0.2}
Val Accuracy: 0.7960

🔍 Trying combination: {'lr': 0.0001, 'batch_size': 32, 'weight_decay': 1e-05, 'dropout': 0.3}
Val Accuracy: 0.7770

 Best Stage 1 parameters: {'lr': 0.0001, 'batch_size': 16, 'weight_decay': 1e-05, 'dropout': 0.2}, Final Val Acc: 0.8269
```

Stage 2

```
🔍 Trying combination: {'lr': 0.001, 'batch_size': 16, 'weight_decay': 0.0001, 'dropout': 0.2}
Epoch 1/5, Val Loss: 7957.6965
Epoch 2/5, Val Loss: 7957.6965
Epoch 3/5, Val Loss: 7957.6965
Epoch 4/5, Val Loss: 7957.6965
Epoch 5/5, Val Loss: 7957.6965
✅ New best combination!

🔍 Trying combination: {'lr': 0.001, 'batch_size': 16, 'weight_decay': 0.0001, 'dropout': 0.3}
Epoch 1/5, Val Loss: 8214.4811
Epoch 2/5, Val Loss: 8214.4811
Epoch 3/5, Val Loss: 8214.4811
Epoch 4/5, Val Loss: 8214.4811
Epoch 5/5, Val Loss: 8214.4811

🔍 Trying combination: {'lr': 0.001, 'batch_size': 16, 'weight_decay': 1e-05, 'dropout': 0.2}
Epoch 1/5, Val Loss: 7932.2832
Epoch 2/5, Val Loss: 7932.2832
Epoch 3/5, Val Loss: 7932.2832
Epoch 4/5, Val Loss: 7932.2832
Epoch 5/5, Val Loss: 7932.2832
✅ New best combination!
```

Conclusion & Future Work

Conclusion

- Developed a two-stage deep learning pipeline for bowel sound analysis:
- Stage 1: Classification of bowel sound types using CNNs.
- Stage 2: Regression for precise start and end time detection of sounds.
- Applied signal preprocessing: filtering, windowing, spectrogram conversion.
- Performed hyper-parameter tuning via grid search for optimal model performance.

Future Work

Model Improvement:
- Explore advanced architectures (e.g., Transformers, attention mechanisms) to enhance feature extraction.

Real-Time Deployment:
- Develop an end-to-end pipeline for real-time bowel sound monitoring and classification on embedded devices or smartphones.
- Optimize model size and inference speed for low-power, edge deployment.

Clinical Integration:
- Collaborate with healthcare professionals to validate models in real-world scenarios.

Annex

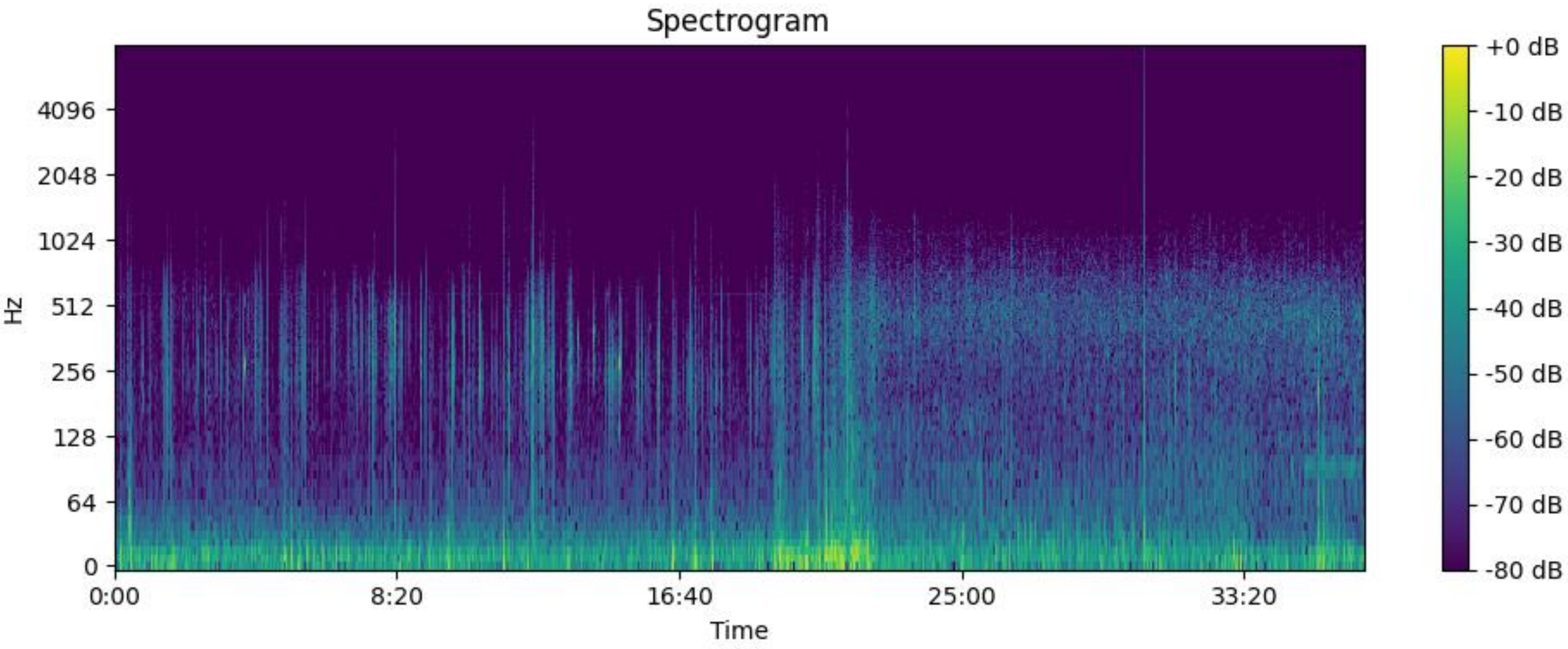
Python library and Framework

- Pytorch.
- Librosa, Pytorchaudio to load audio files.
- Pandas, numpy, scipy.
- SKlearn (Label encoding, dataset split).

GPU resource (Pro Colab Google)

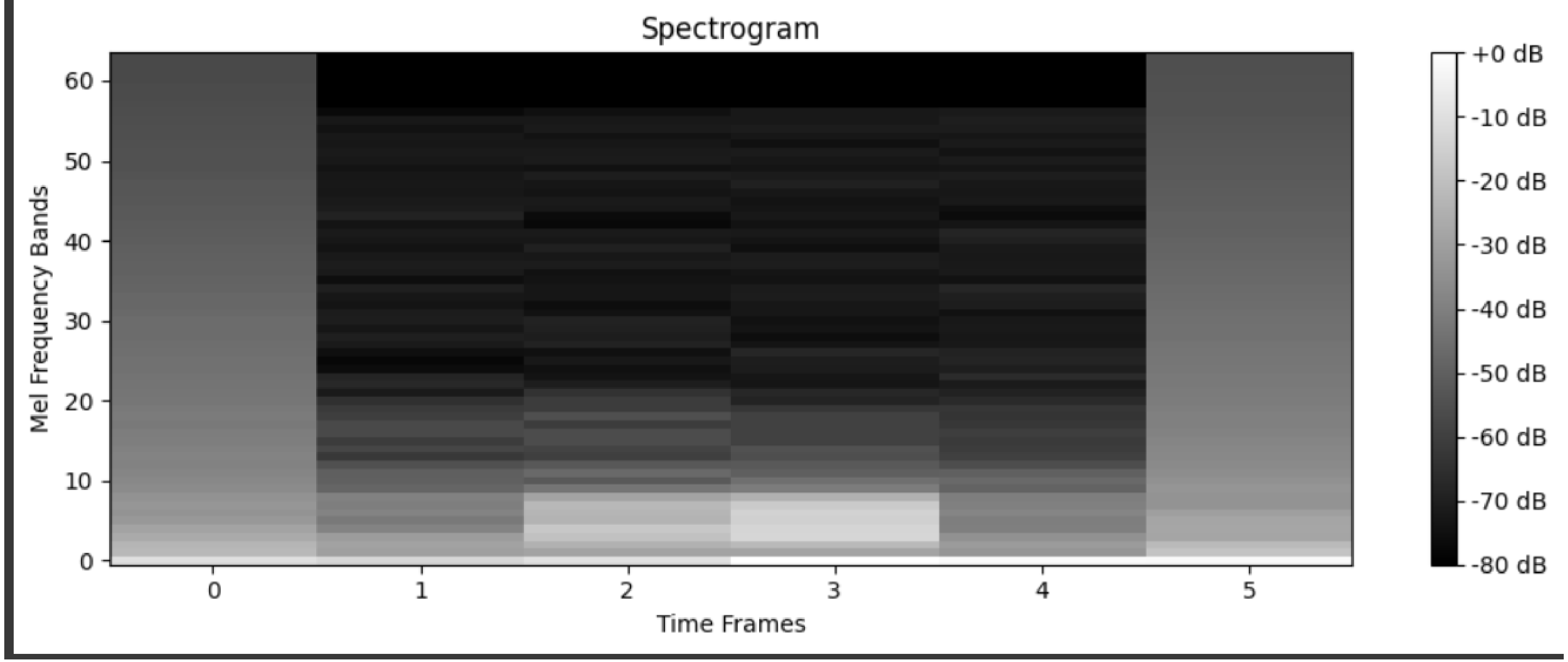
```
Tue Aug 12 17:29:50 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   75C    P0             33W /   70W |   15080MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
+-----------------------------------------------------------------------------------------+
```

Spectrogram (3 channels i.e. RGB image)



1channel (Grayscale)

```
Input Height: 64
Input Width: 11
(64, 11)
```



Simple classifier (old solution)

Data preprocessing pipeline

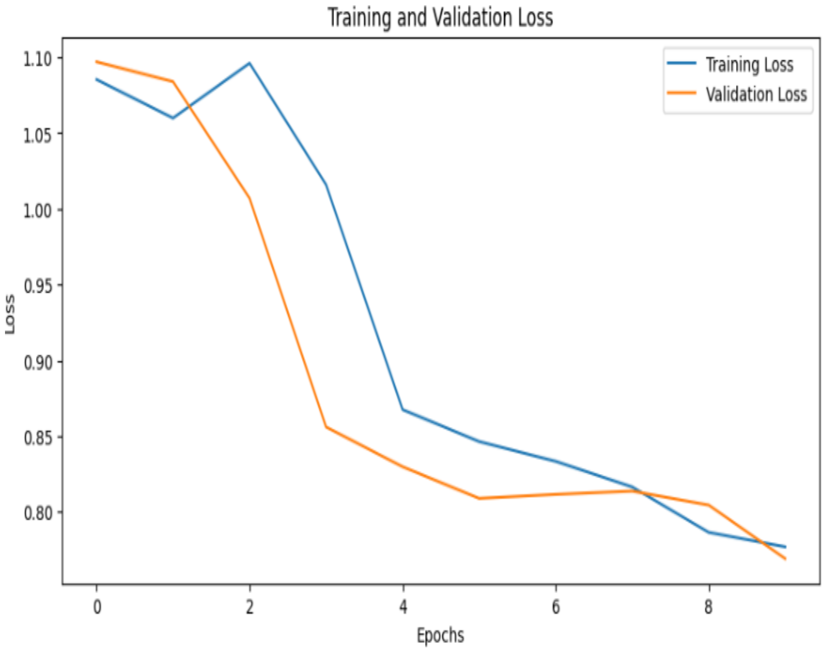
Following a list of procesing done on the dataset before training:

1. Filter to keep only the desired classes: b, mb, and h.
2. Use dropna to detect NaN values.
3. Encode categorical data to numerical (in our case the classes).
4. Visualize the updated class distribution to detect any unbalance between classes.
5. Data augmentation: White Gaussian noise.
6. Create different chunk from the audio file of the classes.
7. Play the audio segments with Ipython.display.
8. Pre-processing audio segments with different durations.
9. Creation of custom dataset classes that map the label to the segment and shuffle the data with the dataloader.
10. Extract the features with Tourchaudio.transform: spectrogram with 1 channel (i.e. grayscale).
11. Split the dataset with sklearn: train, evaluation, test.

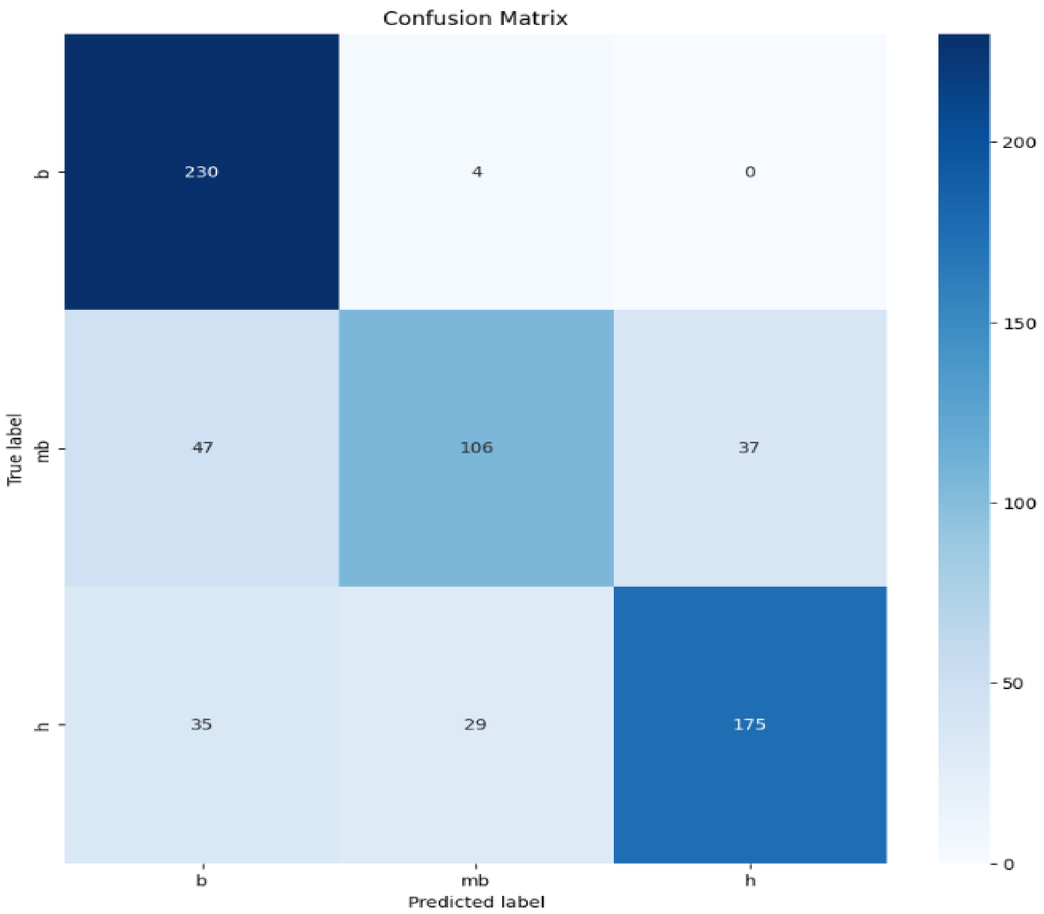
Model and training setup

- Model : CNN
- Chose the hyperparameters: learning rate, batch size, epoches, fft_size, etc.
- Follow the training with tqdm.
- Plot on TensorBoard for the training and evaluation curves (to detect overfitting).
- Cost function: CrossEntropy() and BEntropy() for binary detection.
- Run audio pre-processing on GPU by using Cuda.
- Cloud Platforms for Deep Learning: Google colab Pro.
- Manage code version with Git.
- Choose evaluation metric: use seaborn for the confusion matrix and report (recall, precision, F1, etc.)
- Spectrograms will be utilized as input data, providing a time-frequency representation of audio signals.
- We need to increase the depth of the model

Model and training configuration



	precision	recall	f1-score	support
b	0.74	0.98	0.84	234
mb	0.76	0.56	0.64	190
h	0.83	0.73	0.78	239
accuracy			0.77	663
macro avg	0.78	0.76	0.75	663
weighted avg	0.78	0.77	0.76	663



Still learning until 120 epoch

```
Start Time  End Time  Label WAV_Filename
0.988978  1.242612  h 0_h.wav
1.278083  1.447401  b 1_b.wav
1.496558  1.644029  b 2_b.wav
1.73142 1.851581  b 3_b.wav
4.189271  4.325818  b 4_b.wav
5.106869  5.554744  mb  5_mb.wav
6.013543  6.122781  b 6_b.wav
6.816441  6.931141  b 7_b.wav
7.253392  7.799581  mb  8_mb.wav
7.848738  7.9689  b 9_b.wav
10.546913 10.677998 b 10_b.wav
13.026612 13.42533  mb  11_mb.wav
17.068412 17.215883 b 12_b.wav
17.625525 17.844  mb  13_mb.wav
19.018307 19.160316 b 14_b.wav
20.580408 20.695108 b 15_b.wav
34.082204 34.21329  b 16_b.wav
35.655229 35.775391 b 17_b.wav
40.472618 40.603703 b 18_b.wav
45.929048 46.279443 mb  19_mb.wav
47.332754 47.671391 mb  20_mb.wav
47.780629 48.250352 mb  21_mb.wav
49.353654 49.861784 mb  22_mb.wav
50.0637 50.200247 b 23_b.wav
```

```
Using cuda
Epoch 1
Training: 100%|██████████| 14/14 [00:07<00:00,  1.89batch/s]
loss: 1.7977827787399292
---------------------------
Epoch 2
Training: 100%|██████████| 14/14 [00:06<00:00,  2.00batch/s]
loss: 0.9704123735427856
---------------------------
Epoch 3
Training: 100%|██████████| 14/14 [00:07<00:00,  1.92batch/s]
loss: 0.647476851940155
---------------------------
Epoch 4
Training: 100%|██████████| 14/14 [00:07<00:00,  1.87batch/s]
loss: 0.5368751287460327
---------------------------
Epoch 5
Training: 100%|██████████| 14/14 [00:07<00:00,  1.92batch/s]
loss: 0.5091652274131775
---------------------------
Epoch 6
Training: 100%|██████████| 14/14 [00:07<00:00,  1.84batch/s]
loss: 0.40385910868644714
---------------------------
Epoch 7
Training: 100%|██████████| 14/14 [00:07<00:00,  1.95batch/s]
loss: 0.3975016474723816
---------------------------
Epoch 8
Training: 100%|██████████| 14/14 [00:07<00:00,  1.88batch/s]
```