Problem 1

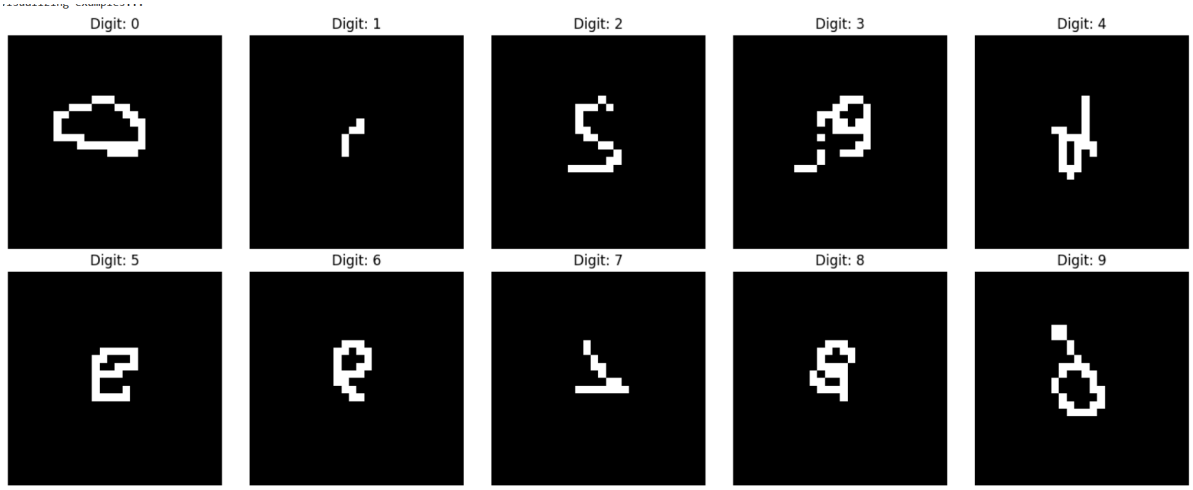
1. Data Collection (10 points)

Follow the data collection procedure in Lab 6. If you did not collect your own gesture data previously, collect it now for digits 0 through 9. Store your recordings in a Google Drive folder and include the link in your submission.

https://drive.google.com/drive/u/0/folders/1aVmIefRN9gRU1zhrmV2ubhVEHiFDSvDC

2. Dataset Preparation and Visualization (10 points)

From the shared Google Drive dataset (link will be posted), create a dataset with 30 examples per digit (0–9), totaling 300 samples. Rasterize each sample and randomly select 1 example per class (0–9) to visualize (10 plots total).



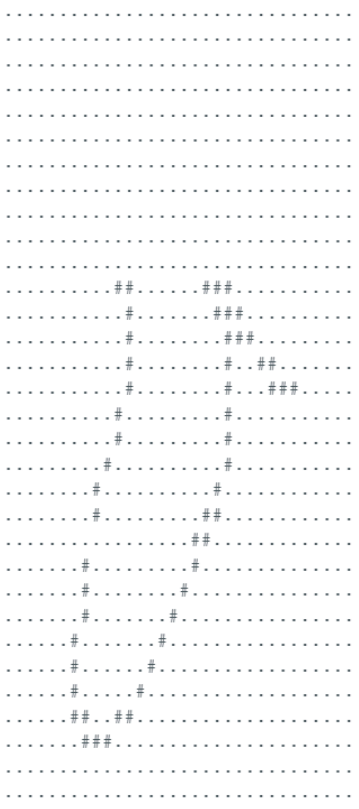
3. Fine-Tuning and Classification Report (15 points)

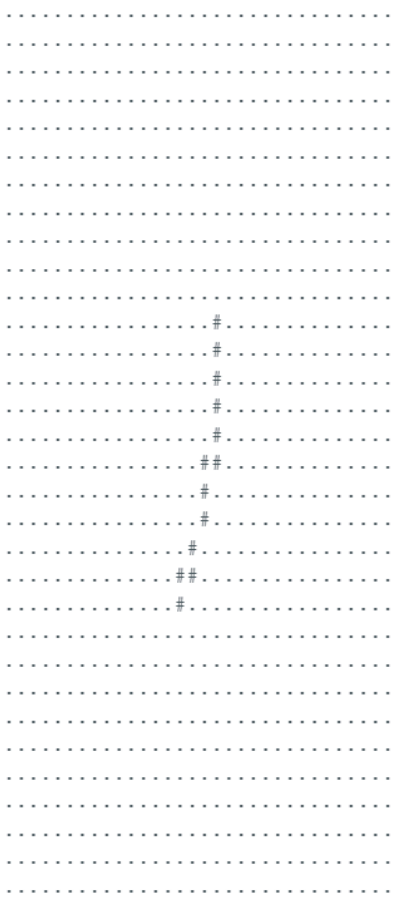
Load the model saved model.keras from Lab 6 and freeze all layers except the final Dense layer. Fine-tune the model on the dataset from part (b). Print the classification report (precision, recall, F1-score, accuracy) on the dataset from part (b) after fine-tuning.

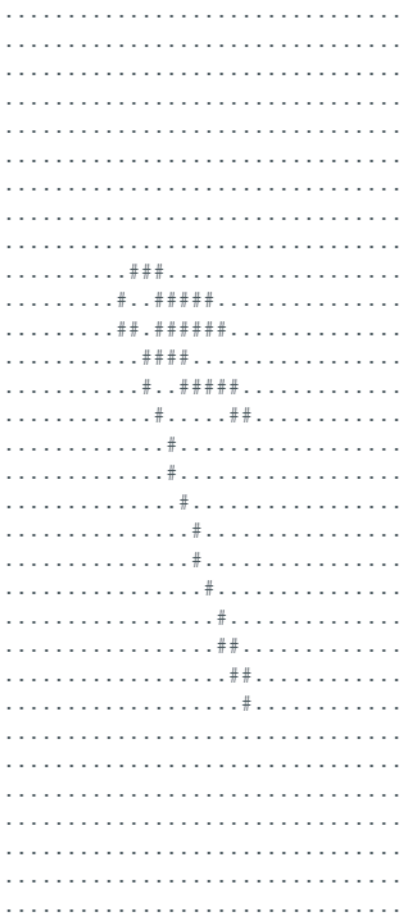
	precision	recall	f1-score	support
0	0.00	0.00	0.00	30
1	0.00	0.00	0.00	30
2	0.00	0.00	0.00	30
3	0.00	0.00	0.00	30
4	0.10	1.00	0.18	30
5	0.00	0.00	0.00	30
6	0.00	0.00	0.00	30
7	0.00	0.00	0.00	30
8	0.00	0.00	0.00	30
9	0.00	0.00	0.00	30
accuracy			0.10	300
macro avg	0.01	0.10	0.02	300
weighted avg	0.01	0.10	0.02	300

4. Deployment and Real-Time Testing (10 points)

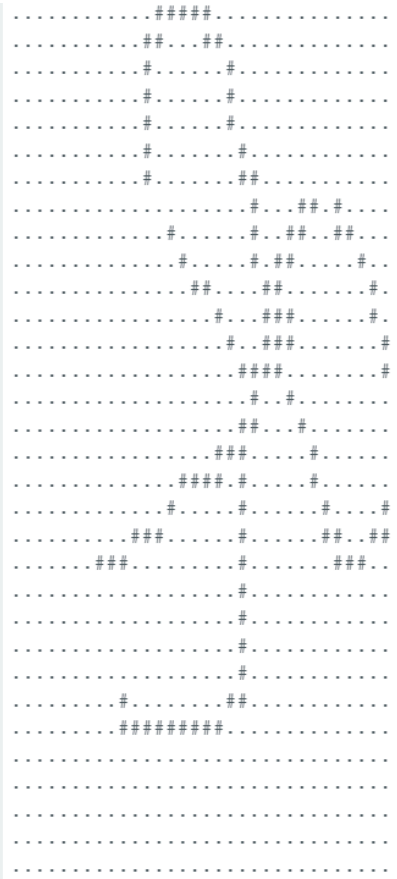
Deploy the fine-tuned model to your Arduino Nano 33 BLE Sense. Submit one screenshot of real-time inference results for each digit (10 total) from the Serial Monitor. Accuracy is not required—just show that the system runs and attempts inference.



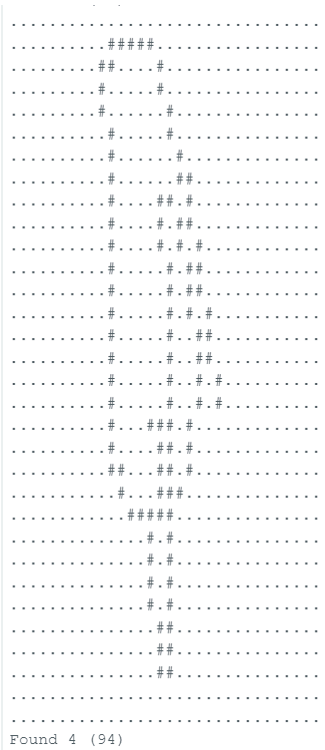




Found 2 (-16)

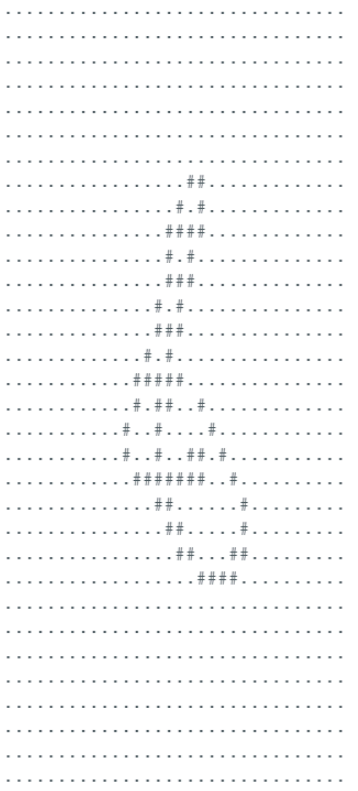


Found 3 (36)

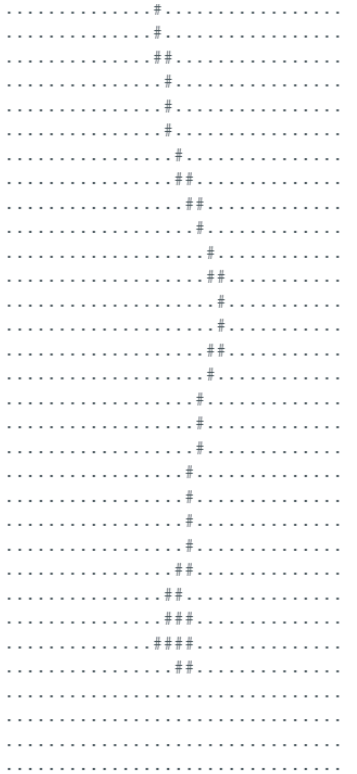


Found 4 (94)

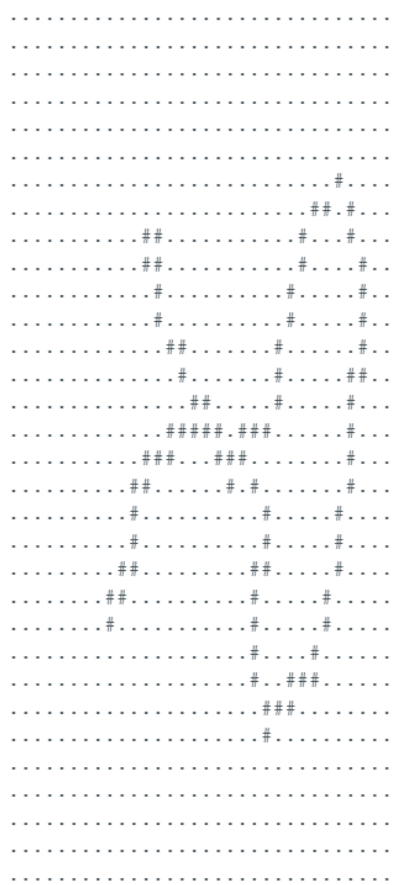
Found 5 (-20)



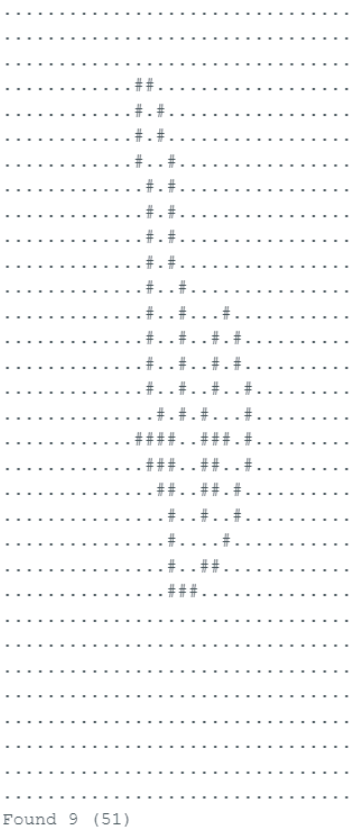
Found 6 (13)



Found 7 (-30)



Found 8 (15)



Found 9 (51)

5. Discussion (5 points)

In a short write-up, reflect on whether real-time results improved after fine-tuning. Why do you think the model works or does not work well? Suggest one or two improvements (e.g., more training data, dropout tuning, etc.).

Problem 2

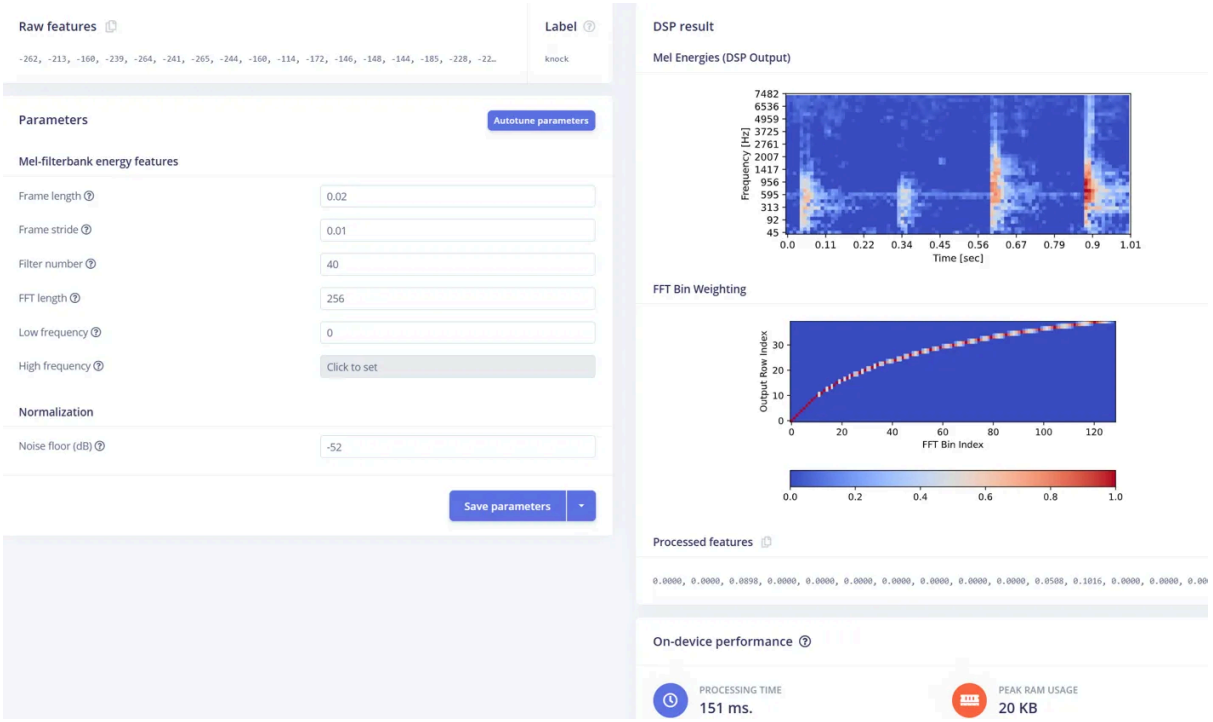
1. Submit a public sharing link to your Edge Impulse project

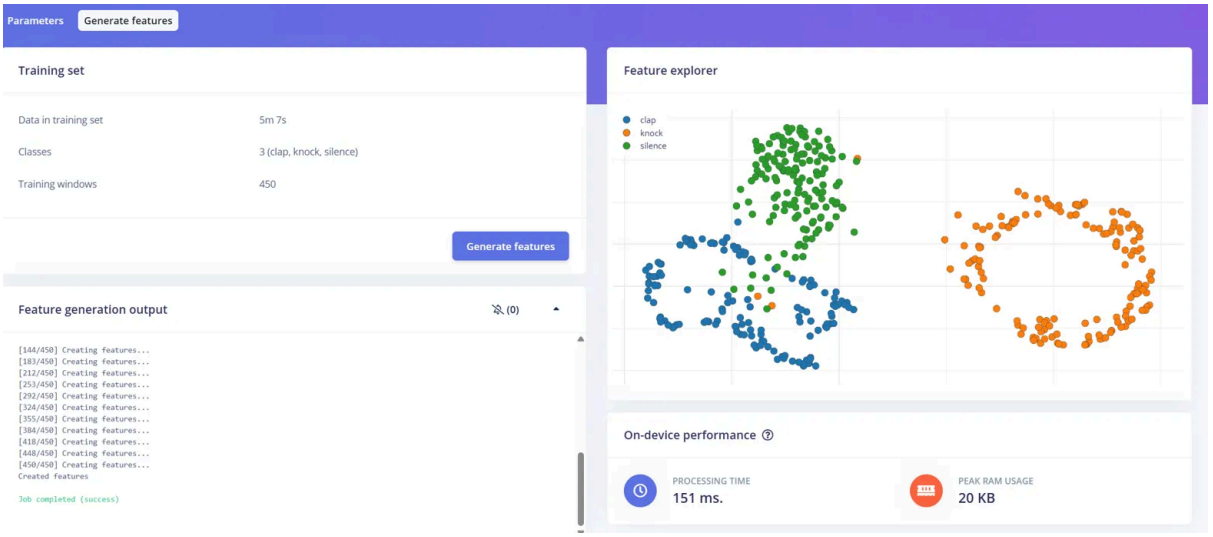
https://studio.edgeimpulse.com/public/711756/live

2. Feature Extraction (10 points)

Use the MFE processing block in Edge Impulse Studio to extract features. Submit screenshots or brief summaries showing the feature extraction step and MFCC settings used.

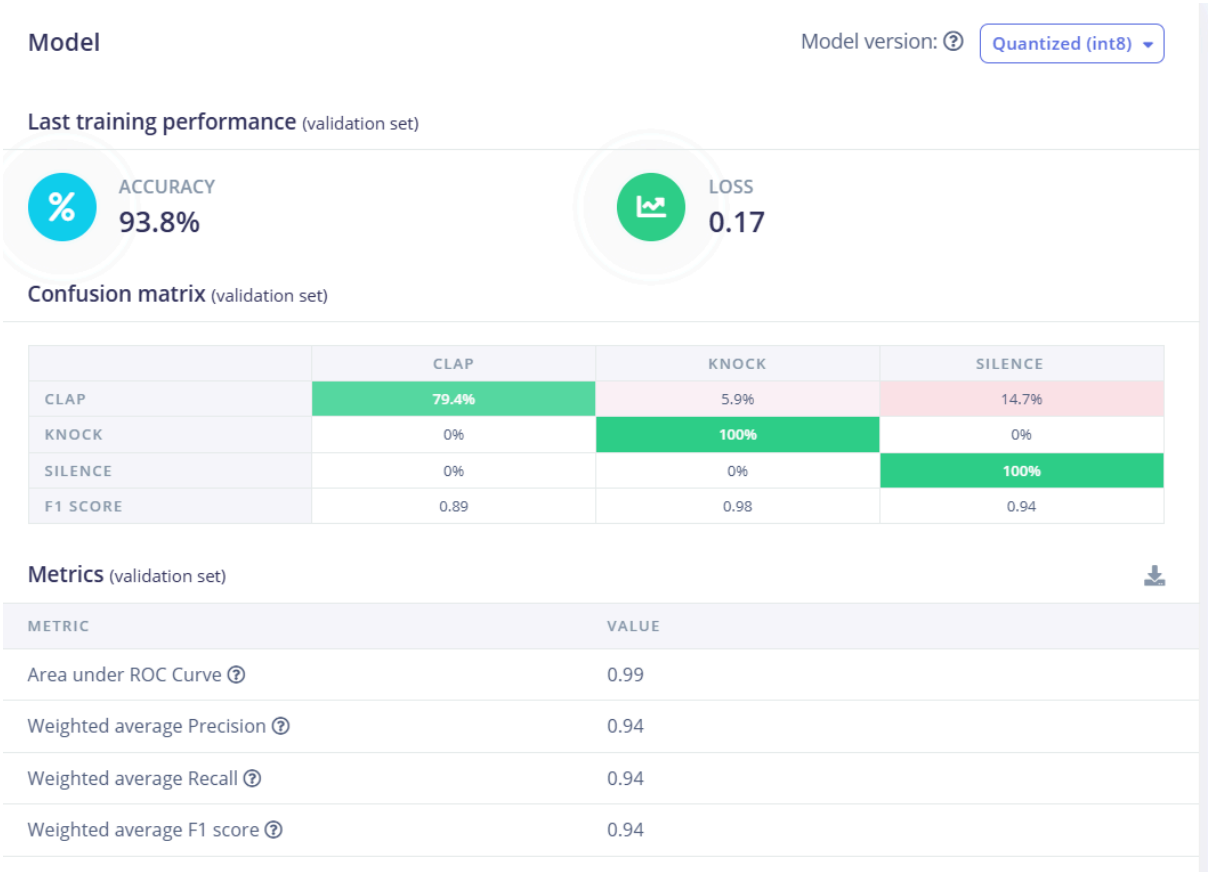
I configured the MFCC processing block with 13 coefficients, 20ms frame length/stride, and 32 mel filters to extract spectral features from the audio data. I used "Autotune parameters" followed by "Generate features" to process 450 training windows across three classes (clap, knock, silence), achieving excellent class separation visible in the feature explorer. The extraction delivered efficient performance with 154ms processing time and 15KB RAM usage, making it suitable for edge deployment while providing strong discriminative features for audio classification.





3. Model Training (10 points)

Use Edge Impulse's Transfer Learning feature with MobileNet 0.1 or a similar small CNN model. Train the model on your dataset and show the test accuracy and confusion matrix.



Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY	LOSS
93.8%	0.17

Confusion matrix (validation set)

	CLAP	KNOCK	SILENCE
CLAP	79.4%	5.9%	14.7%
KNOCK	0%	100%	0%
SILENCE	0%	0%	100%
F1 SCORE	0.89	0.98	0.94

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.94
Weighted average Recall	0.94
Weighted average F1 score	0.94

4. Model Optimization (10 points)

Apply int8 quantization and EON compiler optimizations to reduce model size and improve latency. Submit evidence of model size reduction and show the test accuracy and confusion matrix for the compressed model.

	MFE	TRANSFER LEARNING (KEYWORD...	TOTAL
LATENCY	253 ms.	235 ms.	**488 ms.**
RAM	19.8K	37.1K	**37.1K**
FLASH	-	120.0K	-
ACCURACY			-

	MFE	TRANSFER LEARNING (KEYWORD...	TOTAL
LATENCY	253 ms.	1,610 ms.	**1,863 ms.**
RAM	19.8K	54.9K	**54.9K**
FLASH	-	232.4K	-
ACCURACY			-

5. Deployment and Results (10 points)

Deploy the model to the Arduino Nano 33 BLE Sense using either the Arduino IDE (with TensorFlow Lite for Microcontrollers) or EON compiler method. Submit a screenshot of live inference results from your Arduino Nano 33 BLE Sense for each class (e.g., clap, knock, and silence) from the Serial Monitor or command line.

```
Inferencing settings:
        Interval: 0.06 ms.
        Frame size: 16000
        Sample length: 1000 ms.
        No. of classes: 3
Starting inferencing, press 'b' to break
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 120.602997 ms., Classification: 129.507996 ms., Anomaly: 0ms.):
#Classification results:
    clap: 0.960937
    knock: 0.011719
    silence: 0.027344
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 120.584999 ms., Classification: 129.487000 ms., Anomaly: 0ms.):
#Classification results:
    clap: 0.000000
    knock: 0.996094
    silence: 0.000000
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 120.561996 ms., Classification: 129.399002 ms., Anomaly: 0ms.):
#Classification results:
    clap: 0.425781
    knock: 0.003906
    silence: 0.574219
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 120.667999 ms., Classification: 129.552994 ms., Anomaly: 0ms.):
#Classification results:
    clap: 0.152344
    knock: 0.000000
    silence: 0.843750
```