LIDIA GOMEZ

PROFANITY DETECTOR

The Profanity Detector is a wearable device and application that can recognize when profanity is used in regular speech.

TEAM

LIDIA GOMEZ

Software Developer

INTERESTS

ACADEMIC

- Pursuing a Software Engineering Master's degree at Harvard Extension School.
- Interested in Artificial Intelligence.
- Particularly focused on Machine Learning Models and Deep Neural Networks.

INTERESTS

PERSONAL

- Artistic pursuits - drawing, painting, sculpting, architecture, graphic design
- Writing
- Environmental Protection

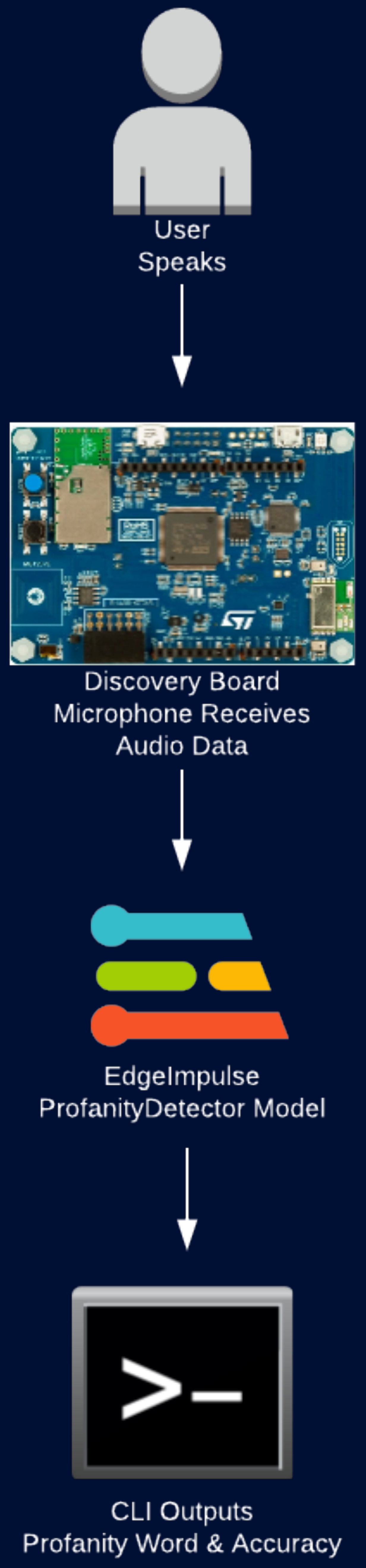
MOTIVATION

- Alejandra Iglesia, a brilliant attorney practicing law in Miami, FL voiced concerns over the ubiquitous use of profanity in her law firm and those they do business with.
- Her own proclivity to use profanity in regular conversation is becoming increasingly difficult to control in professional environments.
- While the use of profanity amongst adults is arguably not directly harmful, our global culture maintains societal standards for communication and decency, specifically in business settings.
- There is currently no marketed wearable device that exclusively detects profanity.

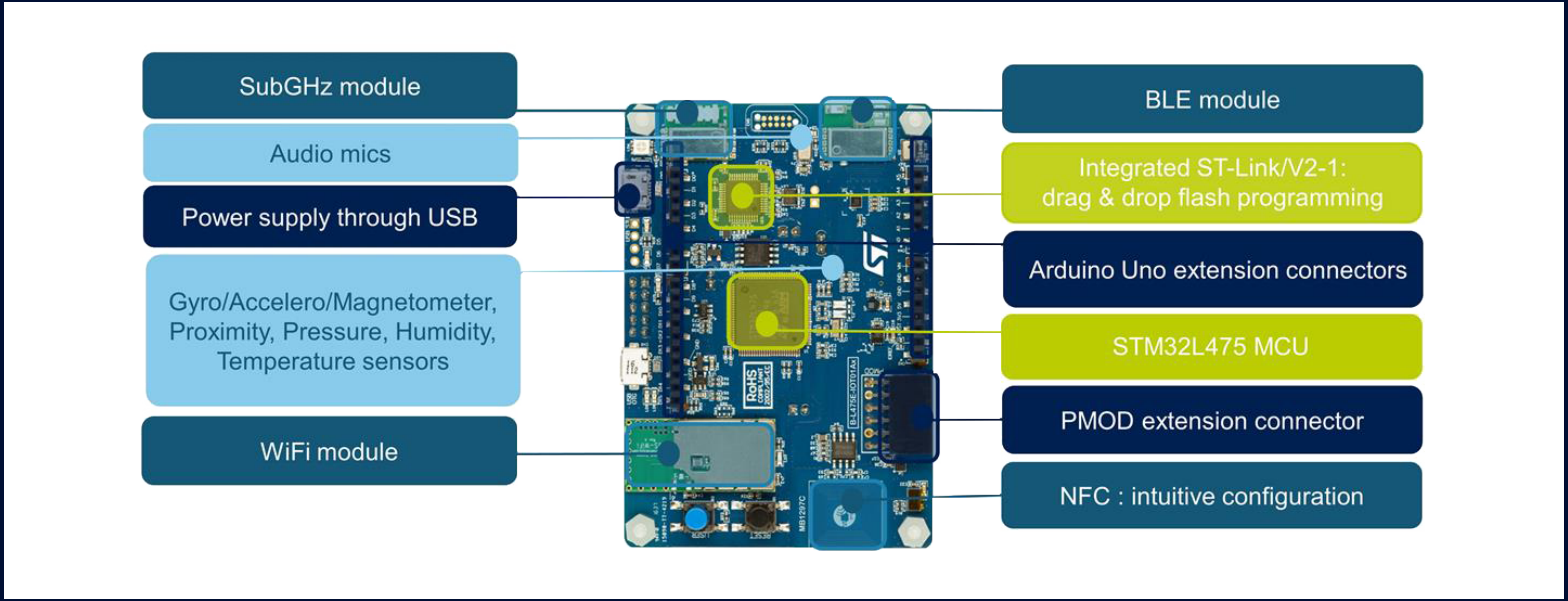
GOAL

The ultimate goal is a model that can correctly recognize when profane words are said in continuous language, and determine the word that is said. An end-product might be a wearable device that uses haptics to vibrate whenever the user says something profane.

The MVP is a model that can correctly recognize when two specific profane words are said ('fuck' and 'bitch'), and distinguish between the words with more than 90% accuracy in real-time.


User Speaks
Discovery Board Microphone Receives Audio Data
EdgeImpulse ProfanityDetector Model
CLI Outputs Profanity Word & Accuracy

REQUIRED EQUIPMENT


SubGHz module
Audio mics
Power supply through USB
Gyro/Accelero/Magnetometer, Proximity, Pressure, Humidity, Temperature sensors
WiFi module
BLE module
Integrated ST-Link/V2-1: drag & drop flash programming
Arduino Uno extension connectors
STM32L475 MCU
PMOD extension connector
NFC : intuitive configuration

STMicroelectronics
B-L475E-IOT01A1

SOFTWARE

EDGE IMPULSE

- Acquire/upload data securely to build datasets.
- Design ML algorithms for classification.
- Test and validate ML model with real-time data.
- Build optimized embedded inference and deploy to device.
- Use model locally on device.

DATA ACQUISITION

Data was collected from two separate sources:

GitHub

theabuseproject / tapad Public

An open dataset consisting of various audio snippets of an extensive list of profane words

LidiaGomez / profanity-detector Public

An proprietary dataset consisting of various audio snippets from different volunteers saying profane words

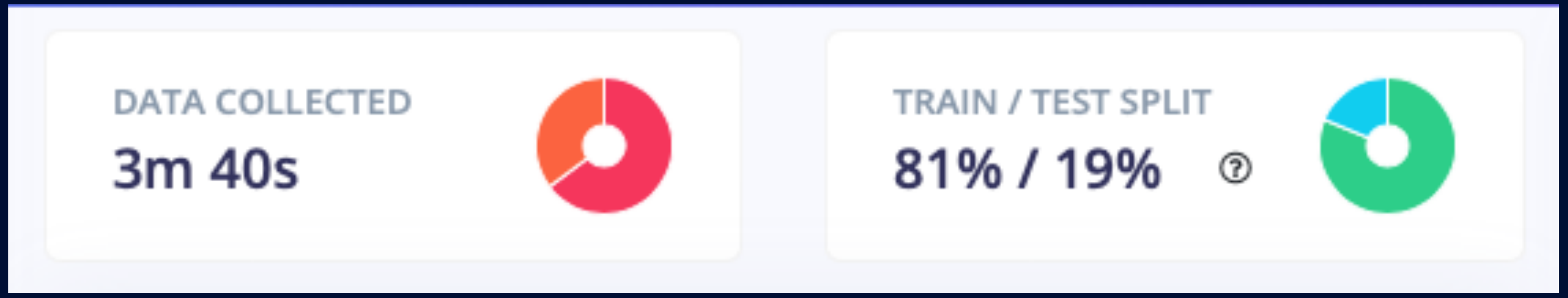
- For proof of concept, the acquisition of data was restricted to two words: ‘fuck’ and ‘bitch’.
- Sets of data were acquired for the target words from the TAPAD dataset.
- Sets of data for the target words were created using volunteers.
- A Python program was written to convert all audio files to .wav files.
- These datasets were uploaded to EdgeImpulse using the EdgeImpulse CLI:

```
$ npm install --g edge-impulse-cli --force

$ edge-impulse-uploader path/to/many/*.wav
```

DATA PROCESSING

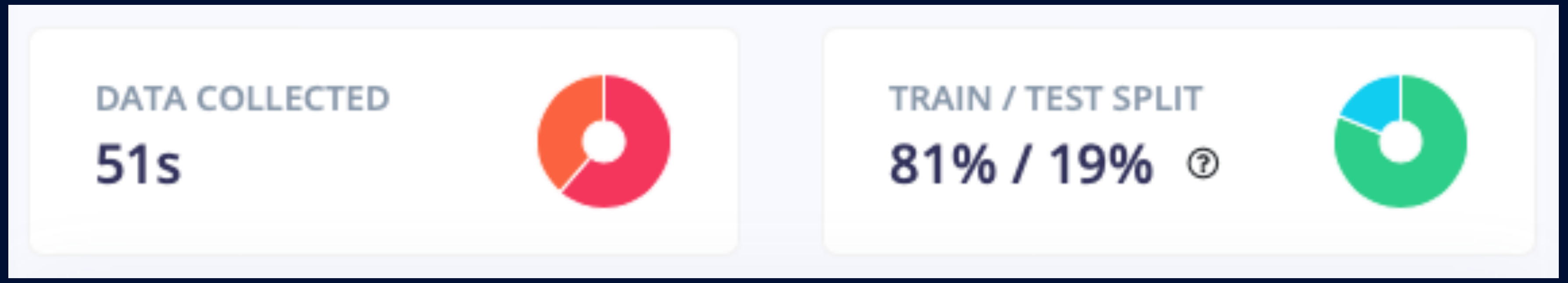
TRAIN DATA



DATA COLLECTED
3m 40s

TRAIN / TEST SPLIT
81% / 19% ⓘ

TEST DATA

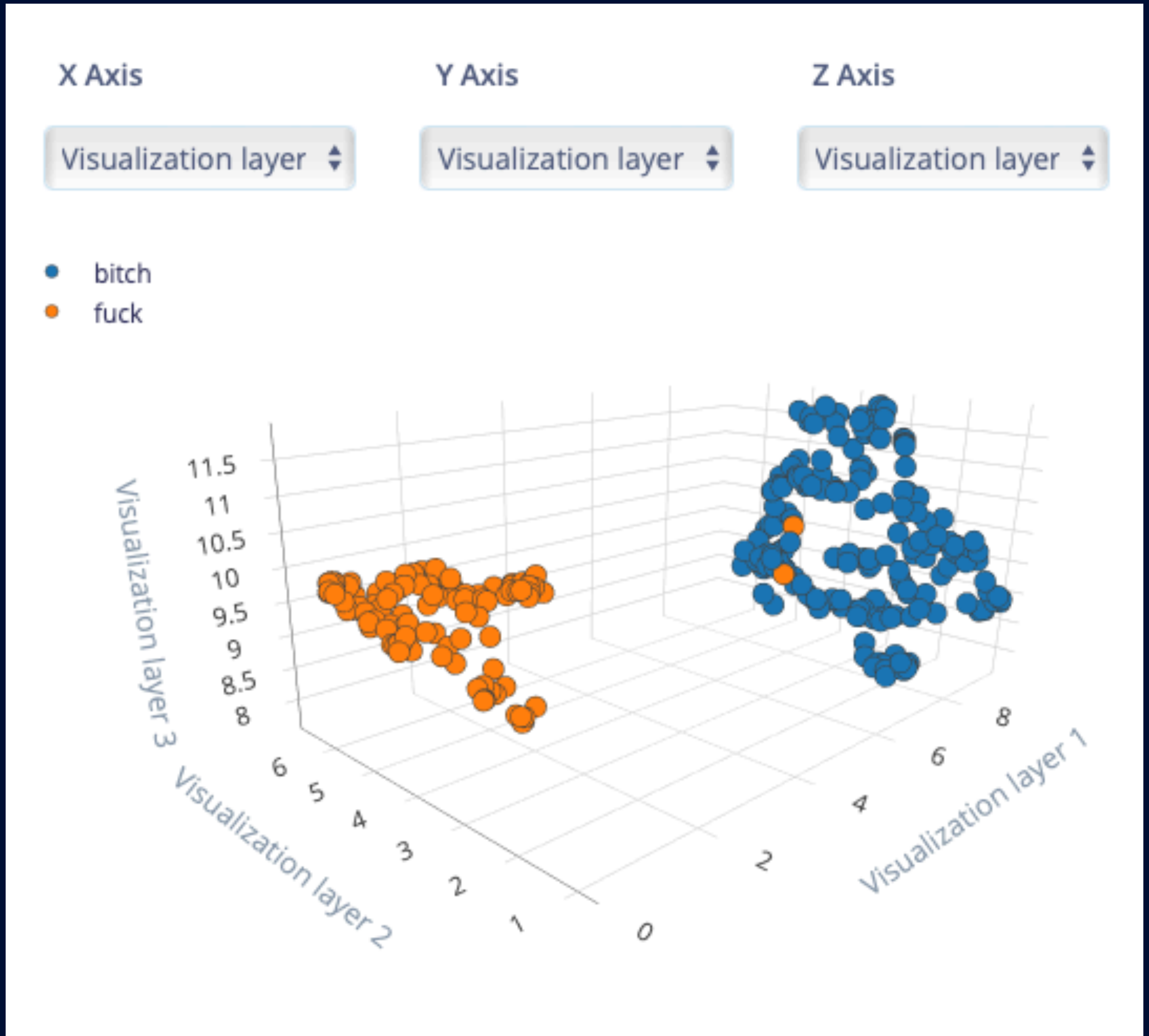


DATA COLLECTED
51s

TRAIN / TEST SPLIT
81% / 19% ⓘ

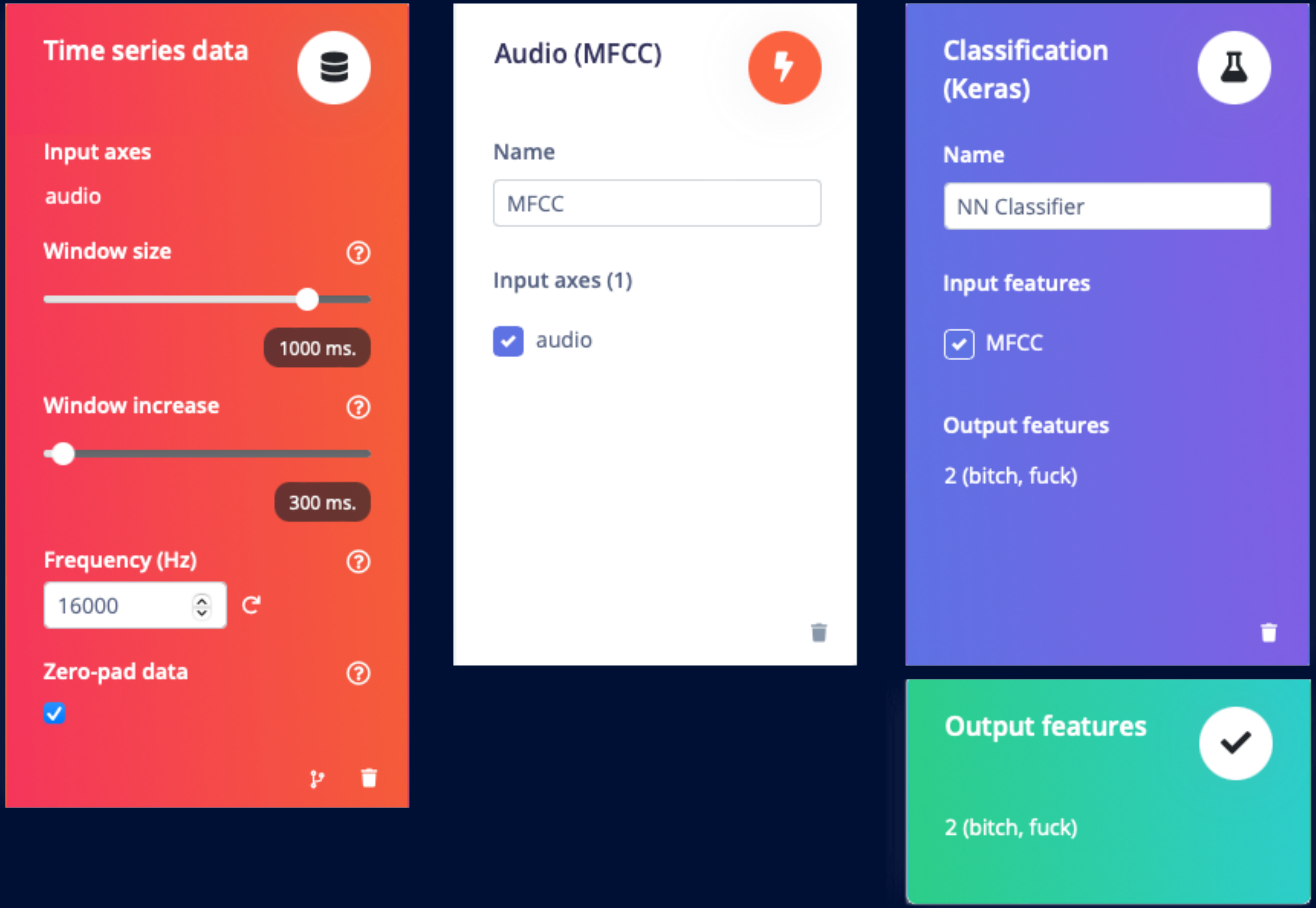
The data collected amounted to a total of 4 minutes and 31 seconds.
While this may not seem like a lot, it must be noted that the average length of each audio file is about 1 second. Thus, 3 minutes and 40 seconds is sufficient to train the network.

FEATURE EXPLORER (313 SAMPLES)



X Axis: Visualization layer
Y Axis: Visualization layer
Z Axis: Visualization layer

MODEL



Time series data

Input axes

audio

Window size

1000 ms.

Window increase

300 ms.

Frequency (Hz)

16000

Zero-pad data

☑

Audio (MFCC)

Name

MFCC

Input axes (1)

☑ audio

Classification (Keras)

Name

NN Classifier

Input features

☑ MFCC

Output features

2 (bitch, fuck)

Output features

2 (bitch, fuck)

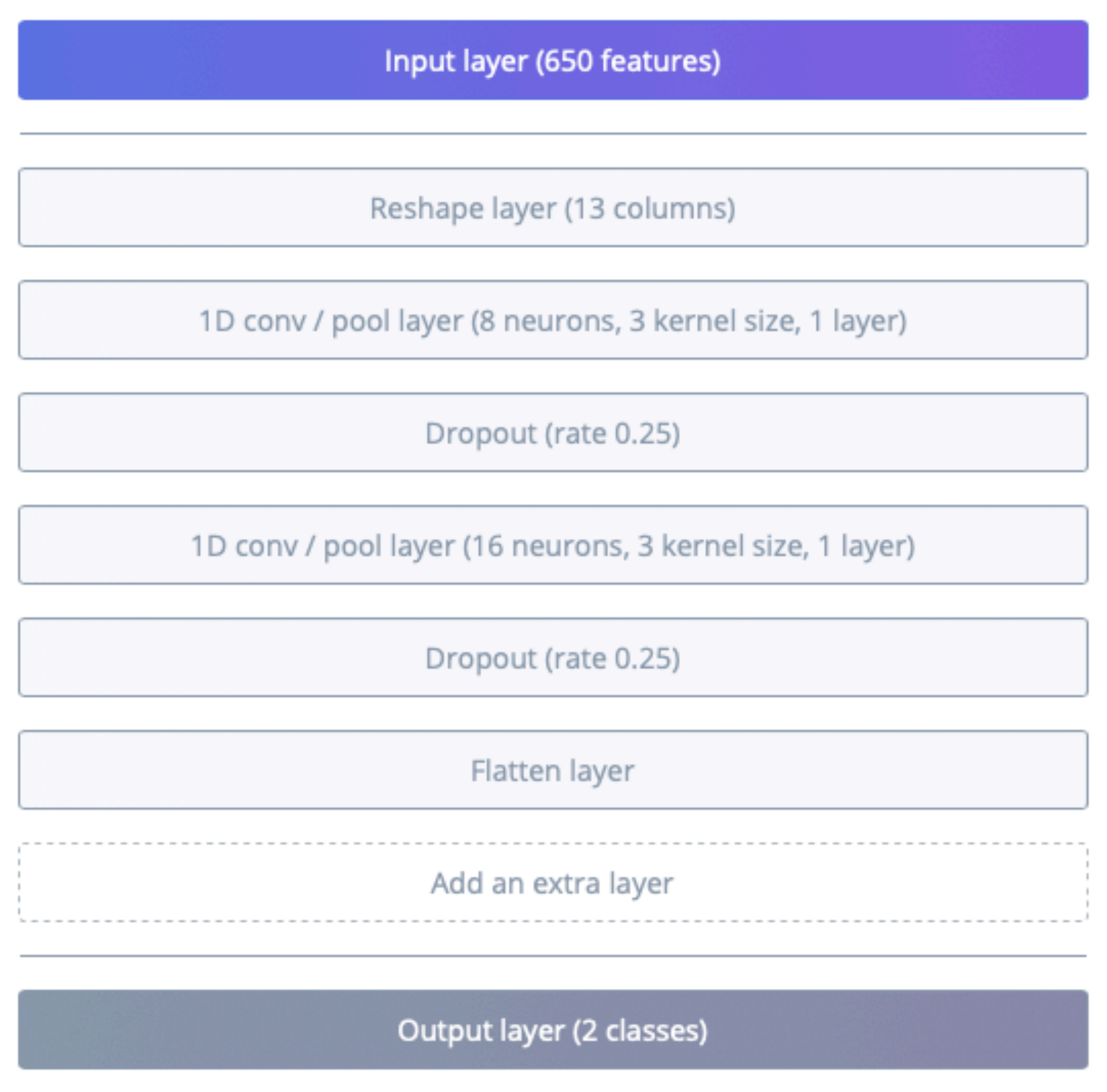
PROCESSING

MFCC is used to extract features from audio signals using Mel Frequency Cepstral Coefficients which work better for human voices than MFE.

Keras is used for classification using the MFCC input features. This works great for audio recognition.

NN CLASSIFIER

NEURAL NETWORK ARCHITECTURE



Input layer (650 features)

Reshape layer (13 columns)

1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

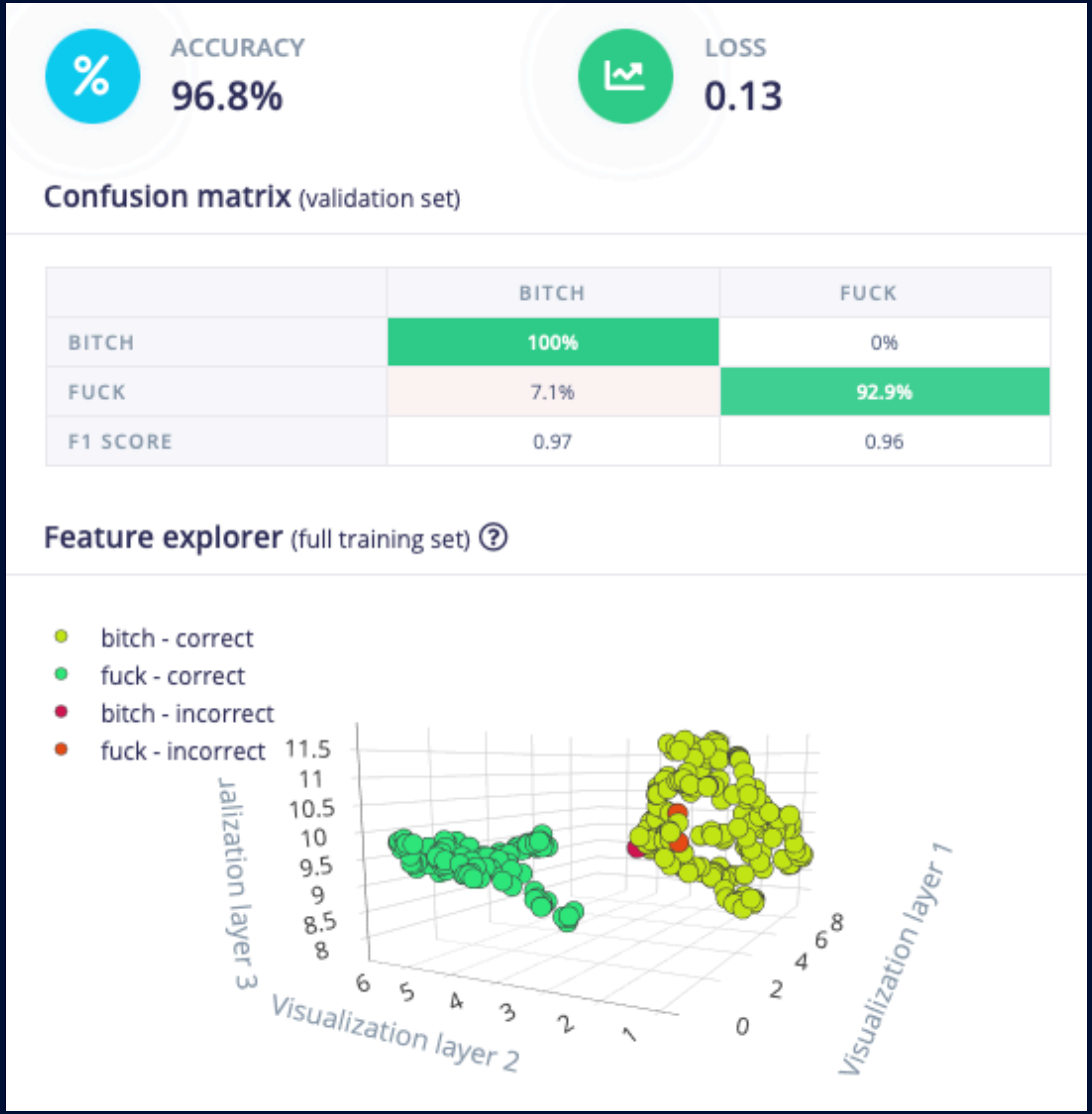
Flatten layer

Add an extra layer

Output layer (2 classes)

The model was trained for 500 epochs, with a learning rate of 0.00005 and a validation set size of 20%.

TRAINING



ACCURACY 96.8%

LOSS 0.13

Confusion matrix (validation set)

	BITCH	FUCK
BITCH	100%	0%
FUCK	7.1%	92.9%
F1 SCORE	0.97	0.96

Feature explorer (full training set) ?

● bitch - correct

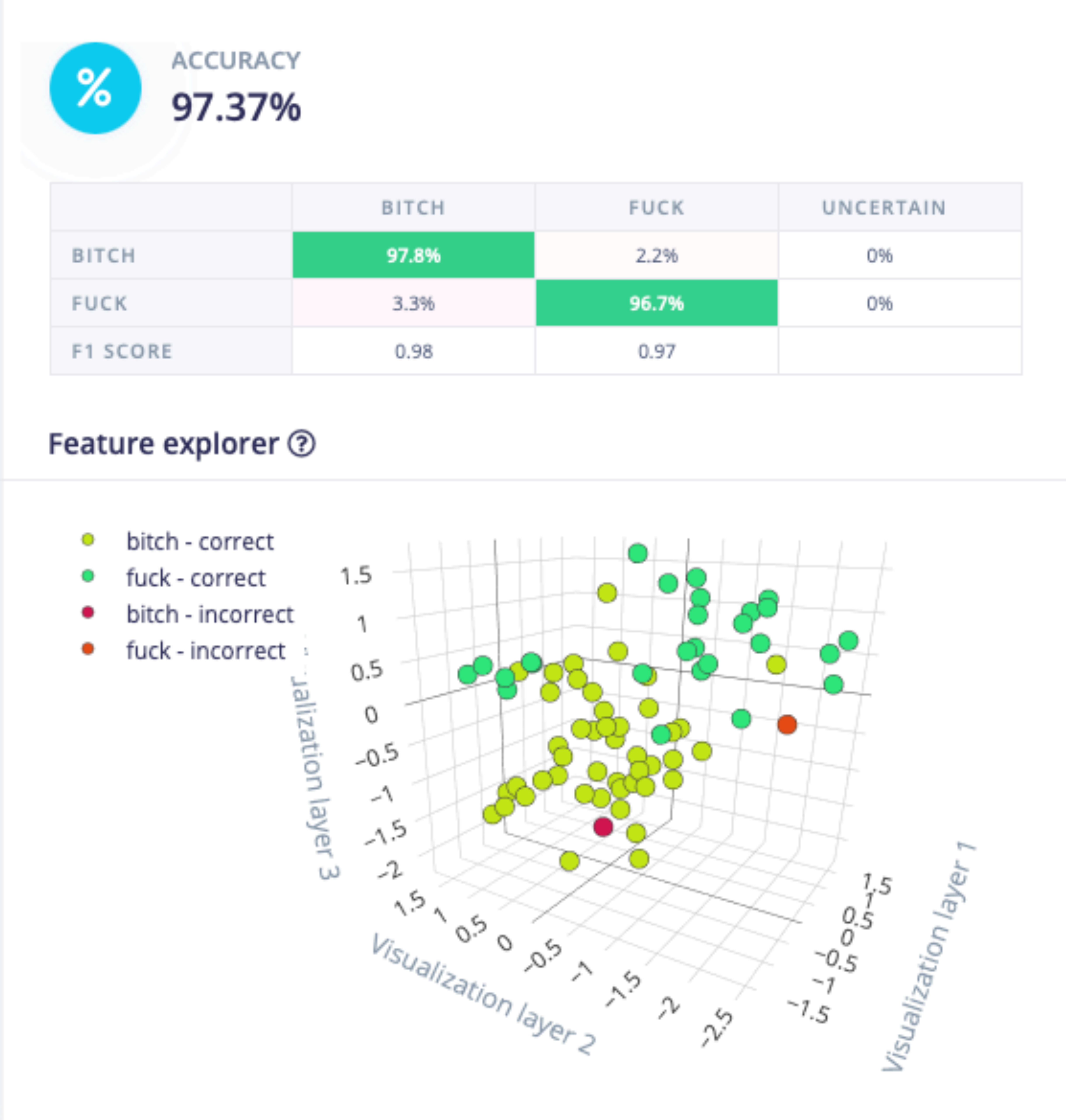
● fuck - correct

● bitch - incorrect

● fuck - incorrect

TESTING

- The model was tested using both the test data and the live classification functionality of EdgeImpulse.
- Testing yielded an accuracy of 97.37%, which is well above the proposed accuracy to meet the initial MVP goal.
- While there were a few misclassified samples for both words, the accuracy result is good enough to deploy the model on the discovery board and complete live testing.



ACCURACY

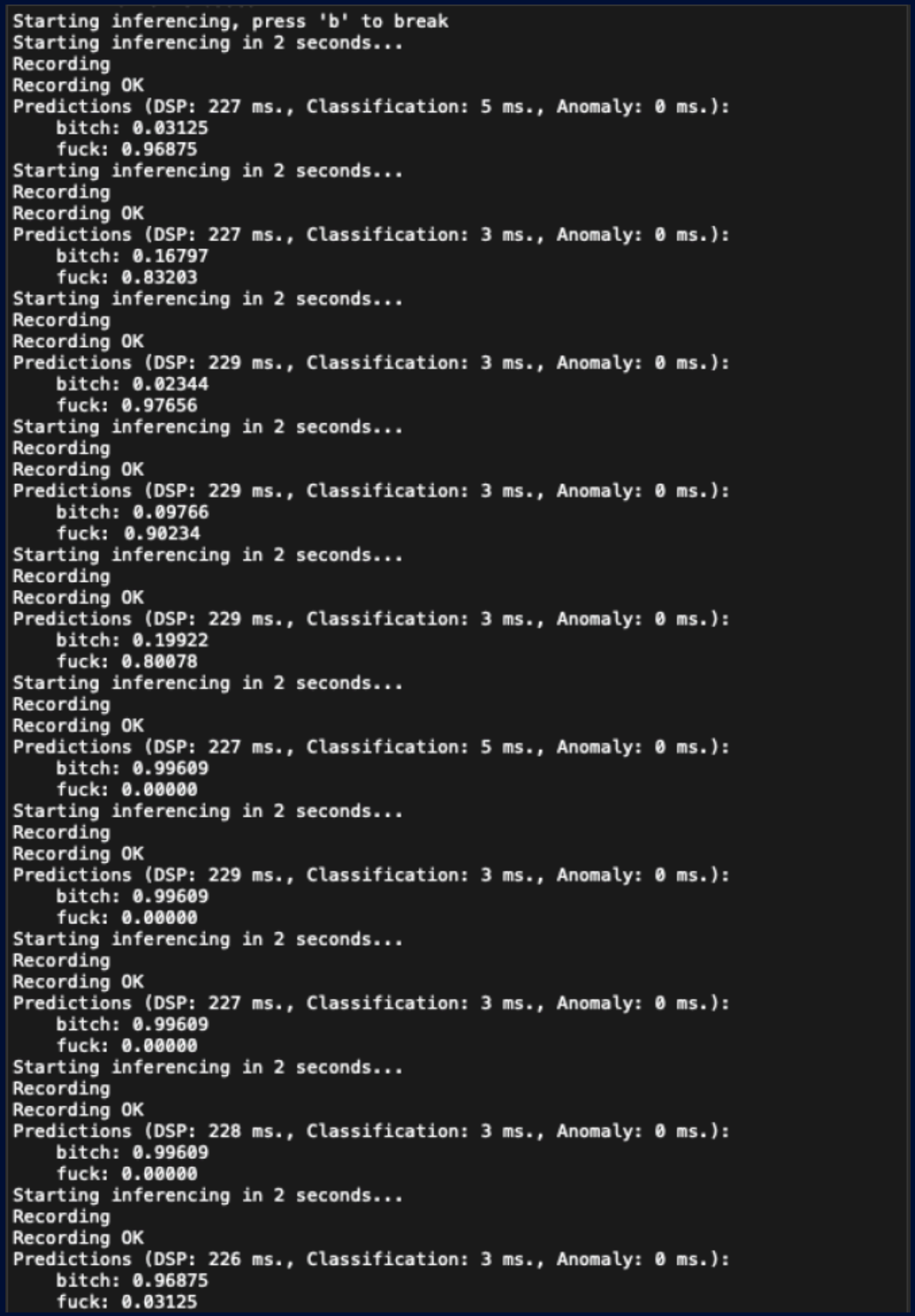
97.37%

	BITCH	FUCK	UNCERTAIN
BITCH	97.8%	2.2%	0%
FUCK	3.3%	96.7%	0%
F1 SCORE	0.98	0.97	

Feature explorer

DEPLOYMENT

- The model was deployed onto the discovery board and tested.
- The WiFi capability of the hardware was used to project the results to the command line.
- During live testing, the words ‘fuck’ and ‘bitch’ were said five times in different intonations.
- Live testing yielded 100% accuracy on almost every iteration of testing.



```
Starting inferencing, press 'b' to break
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 227 ms., Classification: 5 ms., Anomaly: 0 ms.):
    bitch: 0.03125
    fuck: 0.96875
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 227 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.16797
    fuck: 0.83203
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 229 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.02344
    fuck: 0.97656
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 229 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.09766
    fuck: 0.90234
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 229 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.19922
    fuck: 0.80078
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 227 ms., Classification: 5 ms., Anomaly: 0 ms.):
    bitch: 0.99609
    fuck: 0.00000
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 229 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.99609
    fuck: 0.00000
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 227 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.99609
    fuck: 0.00000
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 228 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.99609
    fuck: 0.00000
Starting inferencing in 2 seconds...
Recording
Recording OK
Predictions (DSP: 226 ms., Classification: 3 ms., Anomaly: 0 ms.):
    bitch: 0.96875
    fuck: 0.03125
```

CONCLUSIONS & FUTURE WORK

CONCLUSIONS

The current model correctly distinguishes between two profane words. This is a microcosmic example of the capabilities of using Keras for audio classification.

Although the use of EdgeImpulse provided a complete platform for the training and deployment of this model, it is limited in its capabilities. Furthermore, the use of EdgeImpulse limits the ability of the hardware that may be used.

FUTURE WORK

The aforementioned limitations can be mitigated by foregoing EdgeImpulse and programming the CNN using Python, Librosa Sound Processing Library, Tensorflow, and Keras.

The sound data is processed using Librosa & FFMPEG, transformed from .wav audio files into MFCCs, which are then organized as numpy arrays with which to train the model.

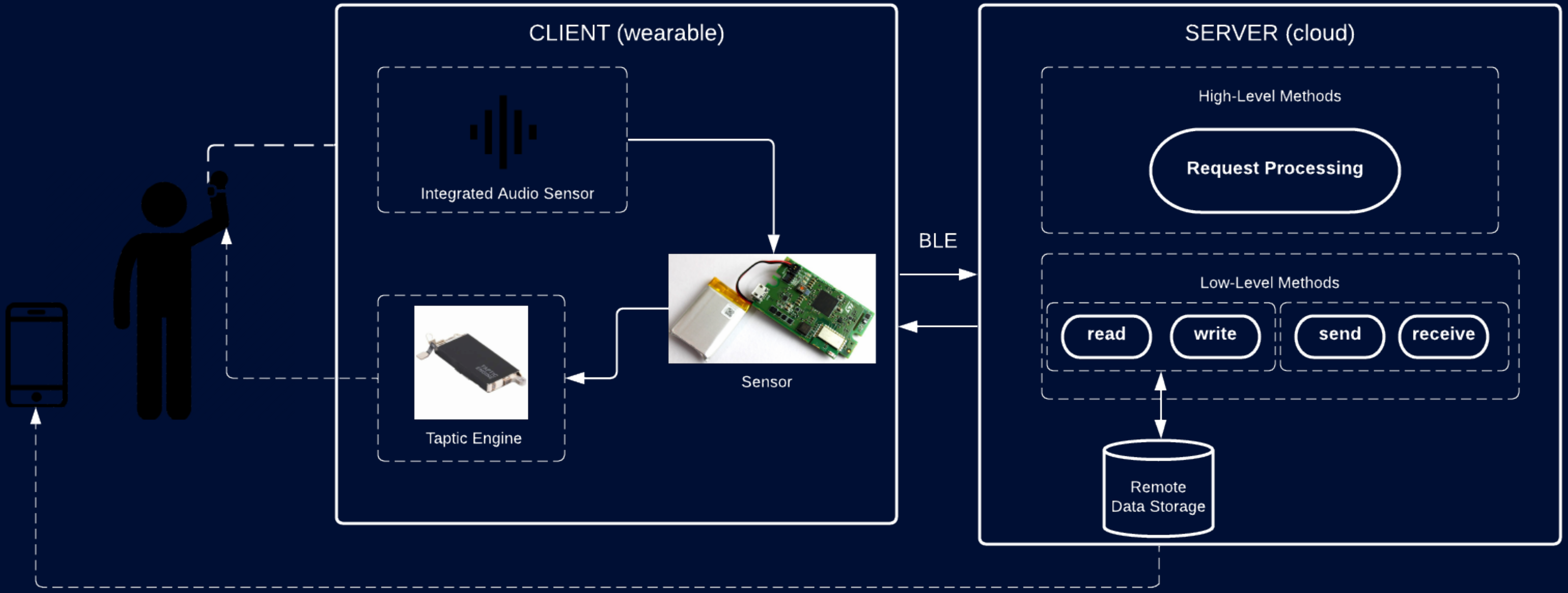
This would provide more flexibility and control in the architecture of the model's layers.

Deployment would no longer be dependent on specific devices, enhancing portability.

CONCLUSIONS & FUTURE WORK

FUTURE ARCHITECTURE DIAGRAM

The current architecture is limited by the platform. However, in the future, cloud services would be leveraged to provide more robust continuous audio processing, recognition, and storage.


CLIENT (wearable)
Integrated Audio Sensor
Sensor
Taptic Engine
BLE
SERVER (cloud)
High-Level Methods
Request Processing
Low-Level Methods
read
write
send
receive
Remote
Data Storage