Assignment 5 - Build and Deploy a Keyword Spotting Model using Edge Impulse

Q. Does the model perform as accurately as expected on your smartphone? List a few methods to improve the model's accuracy.

The model’s performance on the smartphone aligns with expectations but slightly lower compared to the BLE module.

Methods to Improve Model Accuracy

- **Enhance Data Quality and Diversity:**
    - Collect a larger and more diverse dataset to cover various accents, noise levels, and environments.
    - Add data augmentation techniques like noise injection, pitch shifting, or time-stretching to improve the model’s generalization.
- **Optimize Preprocessing Steps:**
    - Experiment with different feature extraction techniques, such as using higher-resolution MFCCs or exploring other audio features like spectrograms.
- **Improve Model Architecture:**
    - Try more advanced lightweight architectures like MobileNet or TinyML-specific models.
    - Add regularization techniques (e.g., dropout) to prevent overfitting during training.
- **Fine-Tune Hyperparameters:**
    - Experiment with learning rates, batch sizes, and epochs to improve the model’s training and testing performance.
- **Post-Training Optimization:**
    - Quantize the model to FP16 or INT8 to make it suitable for deployment without significantly impacting accuracy.
    - Apply techniques like pruning to remove less critical model parameters while maintaining performance.

Q. When building a model for resource-limited hardware, how do you balance fast inference times with acceptable model accuracy? What trade-offs did you encounter?

When building models for resource-limited hardware like the BLE module, there is often a trade-off between achieving faster inference times and maintaining acceptable accuracy. To strike the right balance:

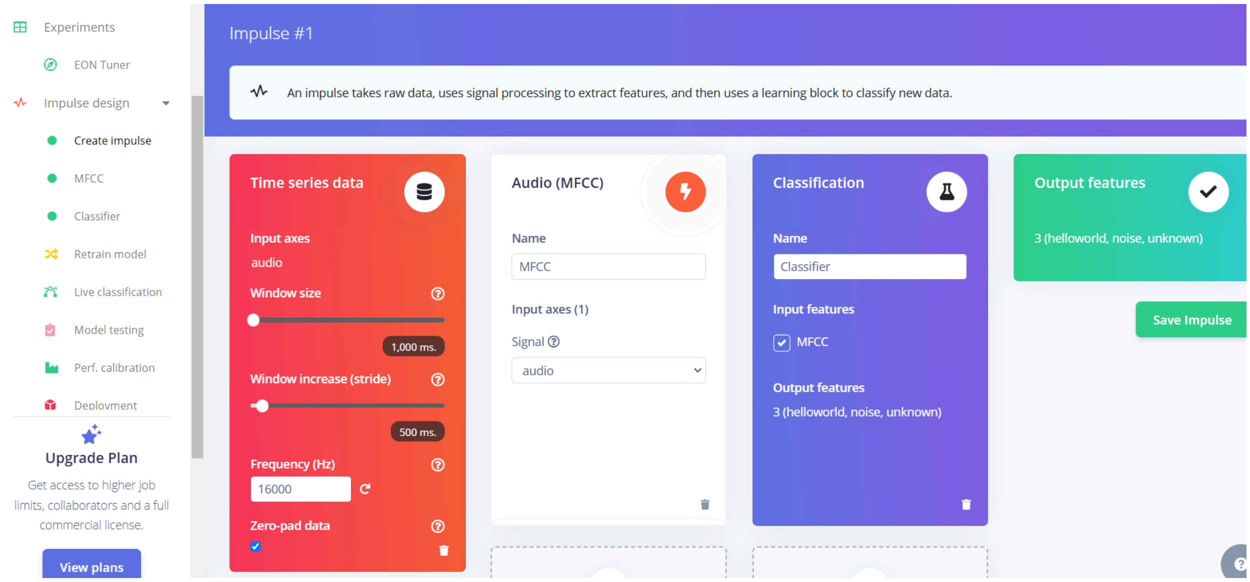
Key Considerations

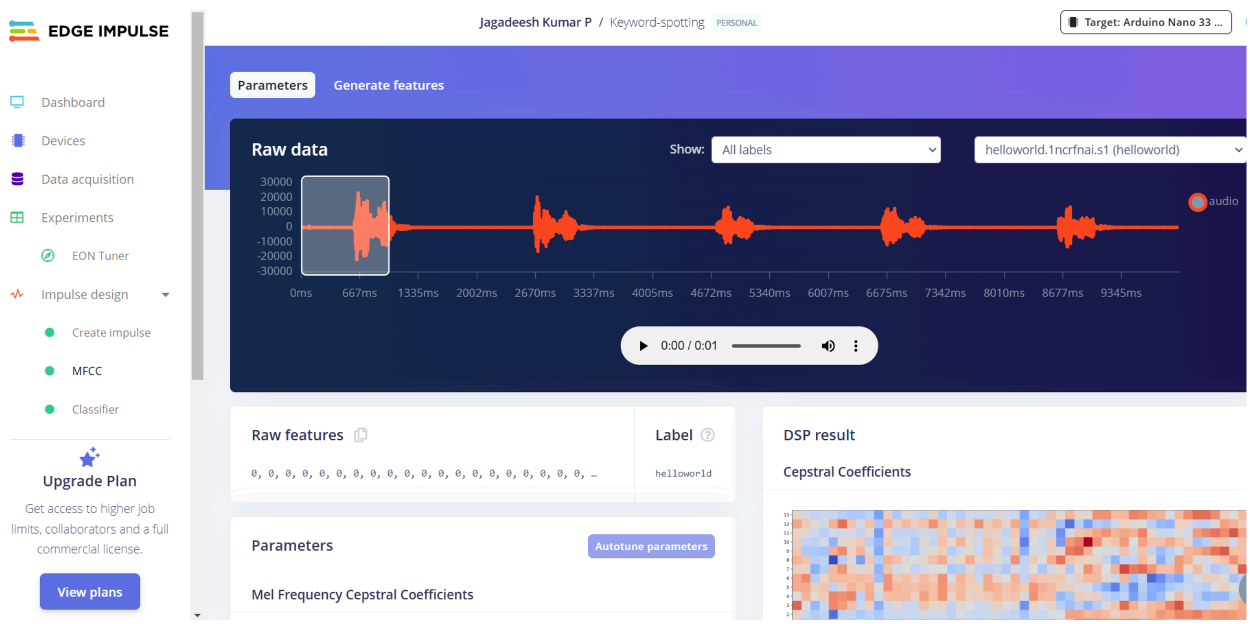
- **Model Size and Complexity:**
    - Smaller models are faster but can compromise accuracy. Selecting architectures optimized for TinyML, such as MobileNet or SqueezeNet, can help strike a balance.
- **Precision Reduction:**
    - Quantization techniques (e.g., INT8 quantization) can significantly improve inference speed and reduce memory usage but may slightly lower accuracy.
- **Feature Simplification:**
    - Reducing the dimensionality of features (e.g., using fewer MFCC coefficients) can speed up inference but might lead to less precise classifications.

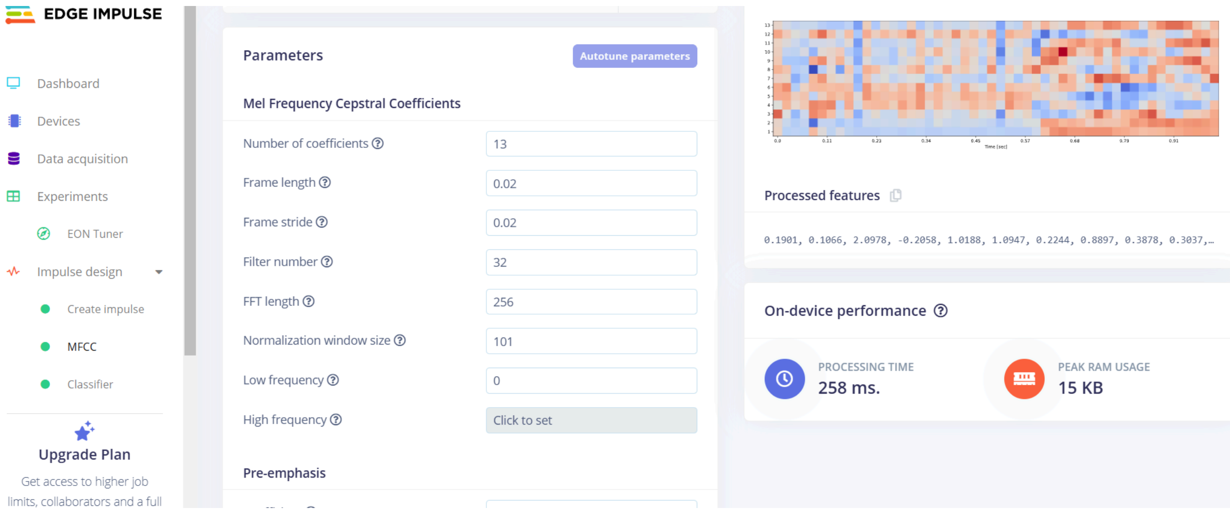
Trade-Offs Encountered

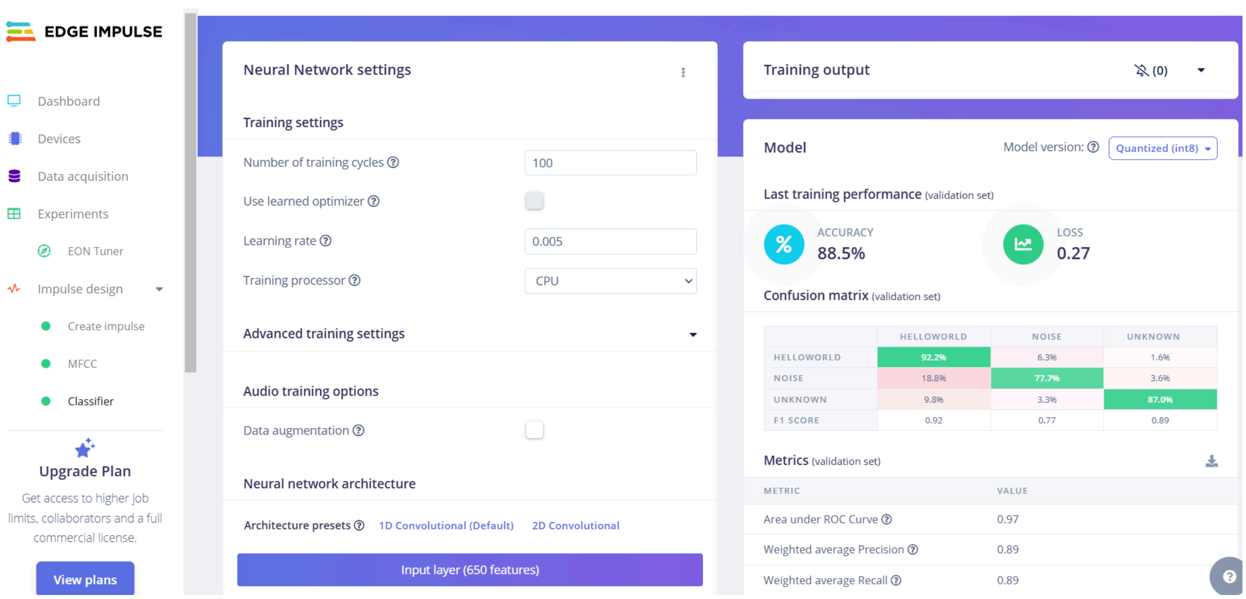
- **Reduced Precision vs. Accuracy:**
    - Lowering the model precision to reduce memory usage and speed up inference might slightly degrade the accuracy.
- **Simplified Features vs. Robustness:**
    - Simplifying features can reduce computation costs but may make the model less robust to variations in the input data.
- **Latency vs. Real-Time Needs:**
    - Optimizing for low latency might require sacrificing some accuracy to meet real-time performance requirements on constrained hardware.

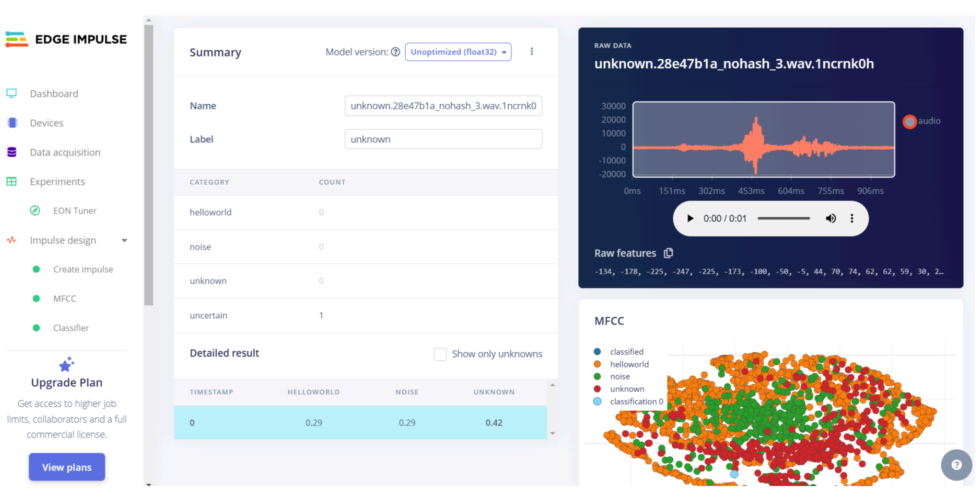
By iteratively testing and validating these trade-offs, the final model can strike a practical balance between fast inference and acceptable accuracy, suitable for the target application.

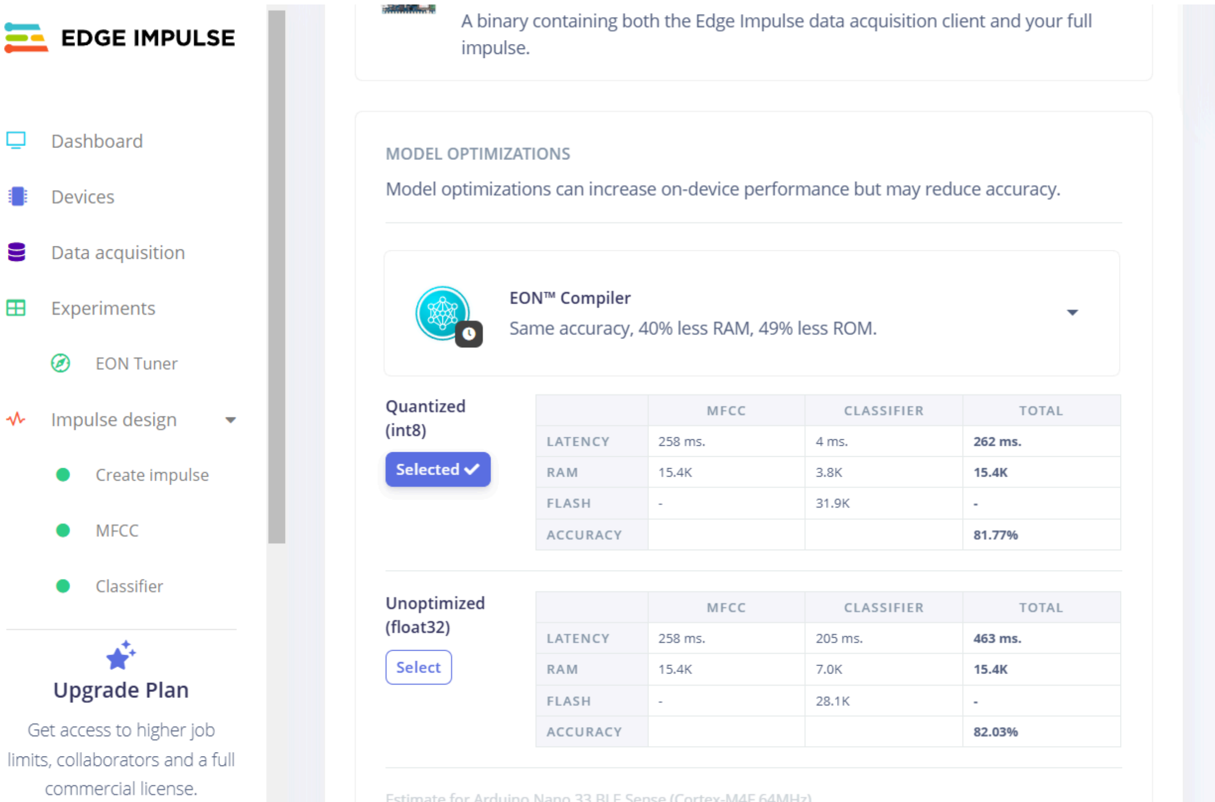

Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Retrain model
Live classification
Model testing
Perf. calibration
Deployment
Upgrade Plan
Get access to higher job limits, collaborators and a full commercial license.
View plans
Impulse #1
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
Time series data
Input axes
audio
Window size
1,000 ms.
Window increase (stride)
500 ms.
Frequency (Hz)
16000
Zero-pad data
Audio (MFCC)
Name
MFCC
Input axes (1)
Signal
audio
Classification
Name
Classifier
Input features
MFCC
Output features
3 (helloworld, noise, unknown)
Output features
3 (helloworld, noise, unknown)
Save Impulse

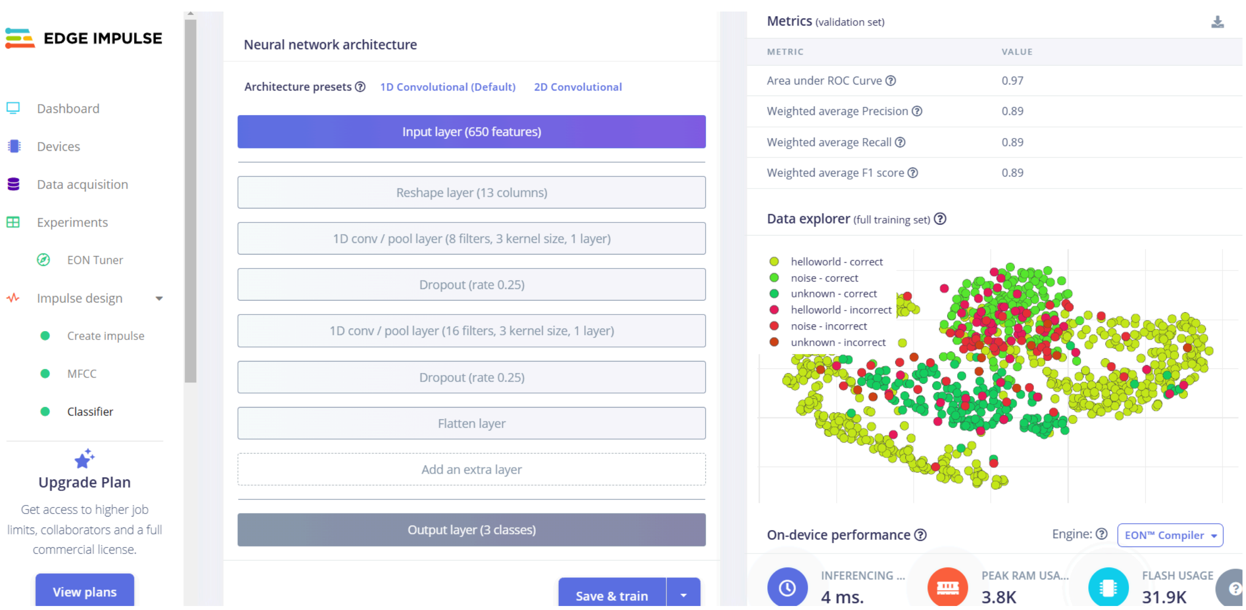

EDGE IMPULSE
Jagadeesh Kumar P / Keyword-spotting PERSONAL
Target: Arduino Nano 33 ...
Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Upgrade Plan
Get access to higher job limits, collaborators and a full commercial license.
View plans
Parameters
Generate features
Raw data
Show:
All labels
helloworld.1ncrfnai.s1 (helloworld)
30000
20000
10000
0
-10000
-20000
-30000
audio
0ms
667ms
1335ms
2002ms
2670ms
3337ms
4005ms
4672ms
5340ms
6007ms
6675ms
7342ms
8010ms
8677ms
9345ms
0:00 / 0:01
Raw features
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
Label
helloworld
DSP result
Cepstral Coefficients
Parameters
Autotune parameters
Mel Frequency Cepstral Coefficients

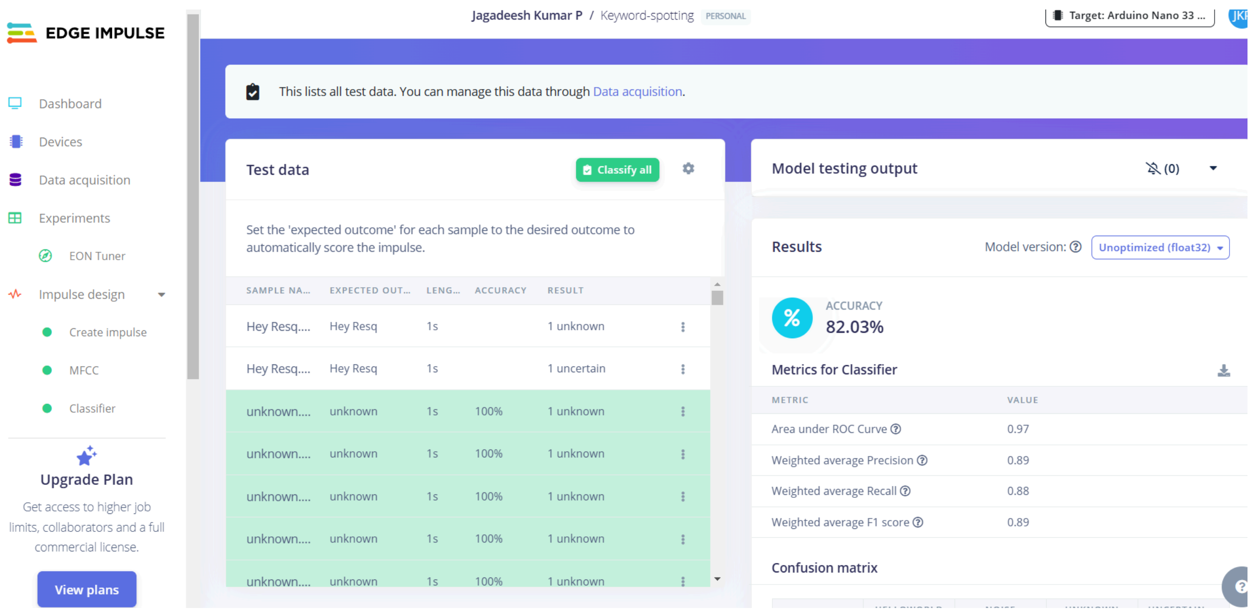

EDGE IMPULSE
Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Upgrade Plan
Get access to higher job limits, collaborators and a full
Parameters
Autotune parameters
Mel Frequency Cepstral Coefficients
Number of coefficients
13
Frame length
0.02
Frame stride
0.02
Filter number
32
FFT length
256
Normalization window size
101
Low frequency
0
High frequency
Click to set
Pre-emphasis
Processed features
0.1901, 0.1066, 2.0978, -0.2058, 1.0188, 1.0947, 0.2244, 0.8897, 0.3878, 0.3037,…
On-device performance
PROCESSING TIME
258 ms.
PEAK RAM USAGE
15 KB

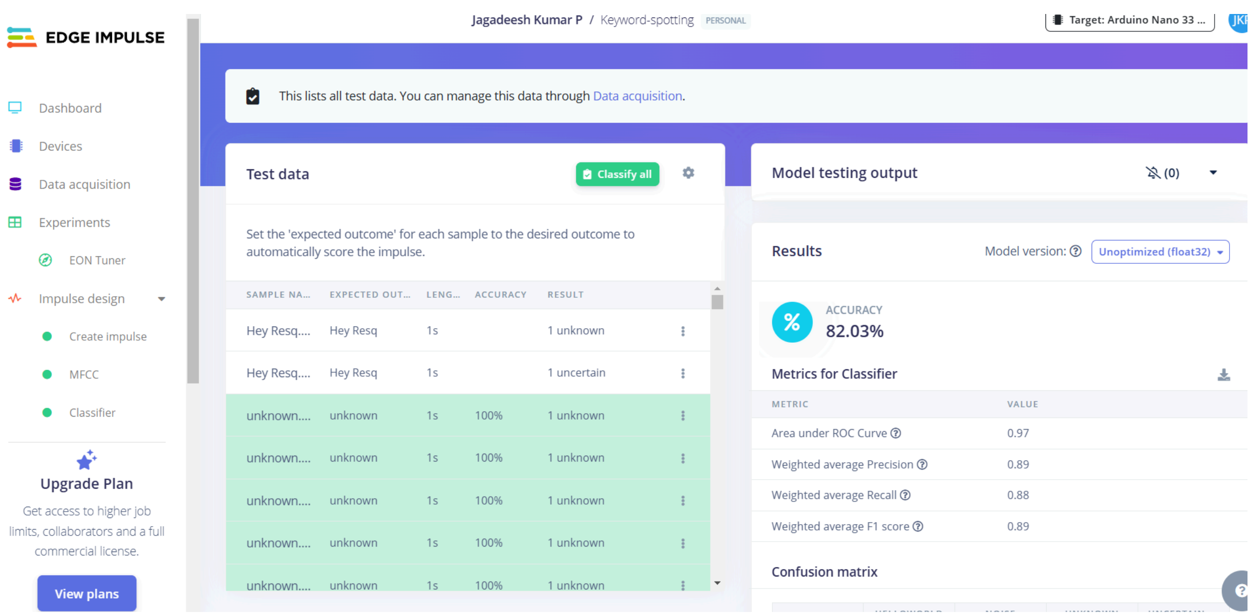

EDGE IMPULSE
Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Upgrade Plan
Get access to higher job limits, collaborators and a full commercial license.
View plans
Neural Network settings
Training settings
Number of training cycles
100
Use learned optimizer
Learning rate
0.005
Training processor
CPU
Advanced training settings
Audio training options
Data augmentation
Neural network architecture
Architecture presets
1D Convolutional (Default)
2D Convolutional
Input layer (650 features)
Training output
(0)
Model
Model version:
Quantized (int8)
Last training performance (validation set)
ACCURACY
88.5%
LOSS
0.27
Confusion matrix (validation set)
HELLOWORLD NOISE UNKNOWN
HELLOWORLD 92.2% 6.3% 1.6%
NOISE 18.8% 77.7% 3.6%
UNKNOWN 9.8% 3.3% 87.0%
F1 SCORE 0.92 0.77 0.89
Metrics (validation set)
METRIC VALUE
Area under ROC Curve 0.97
Weighted average Precision 0.89
Weighted average Recall 0.89


EDGE IMPULSE
Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Upgrade Plan
Get access to higher job limits, collaborators and a full commercial license.
View plans
Summary
Model version:
Unoptimized (float32)
Name
unknown.28e47b1a_nohash_3.wav.1ncrnk0
Label
unknown
CATEGORY COUNT
helloworld 0
noise 0
unknown 0
uncertain 1
Detailed result
Show only unknowns
TIMESTAMP HELLOWORLD NOISE UNKNOWN
0 0.29 0.29 0.42
RAW DATA
unknown.28e47b1a_nohash_3.wav.1ncrnk0h
audio
0:00 / 0:01
Raw features
-134, -178, -225, -247, -225, -173, -100, -50, -5, 44, 70, 74, 62, 62, 59, 30, 2…
MFCC
classified
helloworld
noise
unknown
classification 0


EDGE IMPULSE
Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Upgrade Plan
Get access to higher job limits, collaborators and a full commercial license.
A binary containing both the Edge Impulse data acquisition client and your full impulse.
MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.
EON™ Compiler
Same accuracy, 40% less RAM, 49% less ROM.
Quantized (int8)
Selected
MFCC CLASSIFIER TOTAL
LATENCY 258 ms. 4 ms. 262 ms.
RAM 15.4K 3.8K 15.4K
FLASH - 31.9K -
ACCURACY 81.77%
Unoptimized (float32)
Select
MFCC CLASSIFIER TOTAL
LATENCY 258 ms. 205 ms. 463 ms.
RAM 15.4K 7.0K 15.4K
FLASH - 28.1K -
ACCURACY 82.03%
Estimate for Arduino Nano 33 BLE Sense (Cortex-M4F 64MHz) .

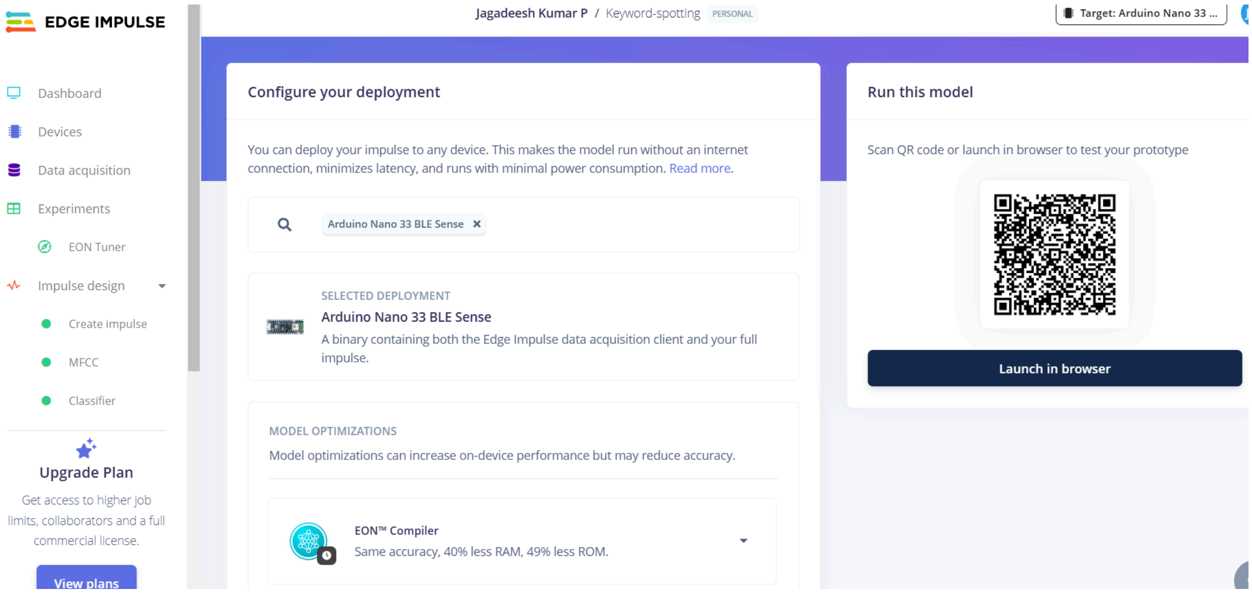

EDGE IMPULSE
Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Create impulse
MFCC
Classifier
Upgrade Plan
Get access to higher job limits, collaborators and a full commercial license.
View plans
Neural network architecture
Architecture presets
1D Convolutional (Default)
2D Convolutional
Input layer (650 features)
Reshape layer (13 columns)
1D conv / pool layer (8 filters, 3 kernel size, 1 layer)
Dropout (rate 0.25)
1D conv / pool layer (16 filters, 3 kernel size, 1 layer)
Dropout (rate 0.25)
Flatten layer
Add an extra layer
Output layer (3 classes)
Save & train
Metrics (validation set)
METRIC VALUE
Area under ROC Curve 0.97
Weighted average Precision 0.89
Weighted average Recall 0.89
Weighted average F1 score 0.89
Data explorer (full training set)
helloworld - correct
noise - correct
unknown - correct
helloworld - incorrect
noise - incorrect
unknown - incorrect
On-device performance
Engine:
EON™ Compiler
INFERENCING ...
4 ms.
PEAK RAM USA...
3.8K
FLASH USAGE
31.9K





Deployment to Smartphone:

- **Process:** The deployment process to the smartphone was fairly straightforward, as Edge Impulse provides an easy-to-use interface to export models to mobile apps. The model was integrated into an app that communicates with the phone's microphone to perform real-time inference.
- **Observations:** The accuracy on the smartphone was good, but not as high as the performance on the Arduino Nano 33 BLE Sense. This might be due to various factors like app limitations, processing power, or the environment in which the test was conducted (e.g., noise, microphone quality).



2. Videos:
   - Record and provide links to:
     - The keyword-spotting model working on your smartphone.
     - The keyword-spotting model working on the embedded Arduino board.

Videos Uploaded

3. Reflections:
   - Share your experience deploying the model to your smartphone and Arduino board. Mention any technical difficulties or interesting observations.

Deployment to Smartphone:

- Process: The deployment process to the smartphone was fairly straightforward, as Edge Impulse provides an easy-to-use interface to export models to mobile apps. The model was integrated into an app that communicates with the phone's microphone to perform real-time inference.

- Observations: The accuracy on the smartphone was good, but not as high as the performance on the Arduino Nano 33 BLE Sense. This might be due to various factors like app limitations, processing power, or the environment in which the test was conducted (e.g., noise, microphone quality).

Deployment to Arduino Nano 33 BLE Sense:

- Process: The deployment to the Arduino board involved converting the model into a format that the board could run (e.g., converting it to a TensorFlow Lite model). Edge Impulse's platform made the conversion process seamless, with easy integration into the Arduino environment.
- Observations: The Arduino Nano 33 BLE Sense performed better with the model in terms of accuracy. This was likely due to the simplicity of the hardware and the focus on low-power, low-latency inference. The BLE Sense board is designed with efficient sensor and signal processing capabilities, which could have contributed to this result.
- Challenges: The main difficulty was ensuring the model size was small enough to fit within the limited memory of the Arduino board. Optimizing the model to run efficiently on such a constrained device was crucial to achieving a good performance.
- Interesting Points: The BLE module’s performance was surprisingly good considering its limited resources. This showed the power of TinyML and the importance of optimizing models for the target hardware.