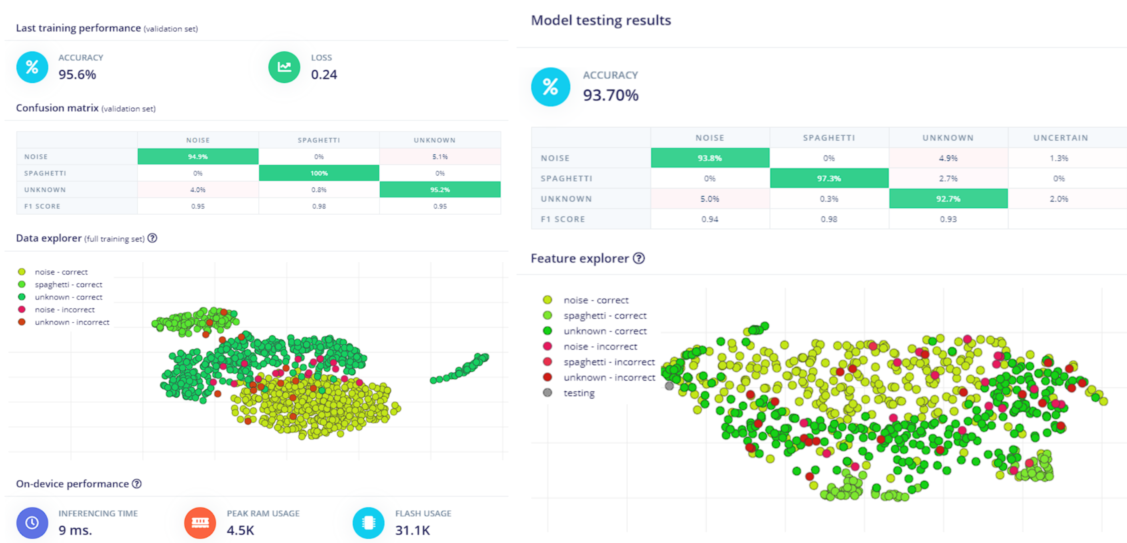
1) The accuracy of the detection can be relatively low at times, but it performs reasonably well in most situations, for the most part marking my keyword as the most likely when its spoken. However, this accuracy could be improved by adding more variations in tone, speed, and variety of voices to the training data. Also, the accuracy would likely be increased if the number of samples per second was increased, as the keyword getting cutoff by the start or end of the recording can cause problems in detection.
2) The main thing to consider is the use case. If you want a model that gives lightning fast response times to input, you probably want to deprioritize accuracy a bit, and if you really want your output to be correct as much as physically possible, you probably want to deprioritize inference speed a bit. Naturally you will need a balance between the two, but where that balance lays is entirely dependent on what you want from the model.



https://drive.google.com/file/d/1Tn-mxr-MK1Csea554z_GqVsKKFqQjyOP/view?usp=sharing

3) We first noticed that when just running `edge-impulse-run-impulse`, we were almost entirely unable to get the model to recognize our keywords, but when we switched to `edge-impulse-run-impulse --continuous`, the results, while far from perfect, were much better. This goes to show how important the number of samples is, as few samples per second is relatively unlikely to pick up a centered snippet of your keyword. Another surprising thing that we noticed was that when we played back a snippet of audio from our phones of us repeating the keyword, we actually retrieved better results than we got when we recorded the original snippets. This one seems particularly hard to explain, but it may have something to do with the digitized sound being processed differently by the model, or maybe something simple like volume, distance, and/or coincidence.

https://drive.google.com/file/d/1vEBPEe35crTwxK8bT3XPFvw_4SbqXzdB/view?usp=sharing