DEEPCRAFT™ Ready Model for Siren Detection

Introduction

In this document, we describe the DEEPCRAFT™ Ready Model for Siren Detection, an audio-based AI model developed by Imagimob, an Infineon Technologies company, that detects sirens from emergency vehicles. We provide details about the technical specifications of this machine learning model, its performance in common scenarios, and various test results for the model including the real-time testing on an Infineon PSOC™ 6 board.

Background

The siren is one of the most essential components of emergency vehicles to alert other drivers and pedestrians regarding the urgency of their task. However, environmental noise and/or headphones might prevent people from detecting the presence of an emergency vehicle. In such situations, emergency vehicles may face additional delays and sometimes even collisions due to inappropriate communication. Therefore, it is essential to efficiently detect the emergency vehicle siren sound to provide notifications or actions to pedestrians. For example, one possible use case is to switch off the noise canceling in headphones. When out in the city, people might use noise canceling when commuting, shopping, and moving in noisy environments with traffic, crowds, etc. However, when an emergency vehicle approaches a street, you'd want to be attentive both for your safety and to be able to leave space for the vehicle.

In this document, we solve the emergency vehicle siren detection problem by employing an audio-based AI model. The model's primary goal is to detect the presence of a siren, regardless of the emergency vehicle type, such as a police car, ambulance, or fire truck. Moreover, the model is trained to detect sirens from standard U.S. emergency vehicles. Finally, the target group of this model is pedestrians only. Even though such a model is applicable in the automotive sector to inform the drivers for upcoming emergency vehicles, deploying the model in the automotive section poses significant challenges. Therefore, it is out of the scope of this version.

The DEEPCRAFT™ Ready Model for Siren Detection is trained on data from U.S.-based sirens. To increase the robustness of the model, we exploited a wide range of indoor and outdoor non-siren sounds. The model employs several convolutional neural networks (CNNs) to capture the information from the raw data, and it is

trained using DEEPCRAFT™ Studio. It is designed to operate on an Infineon PSOC™ 6 hardware with IoT sense expansion kit (CY8CKIT-028-SENSE).

To evaluate the performance of our model, we performed an extensive analysis on realistic datasets. Moreover, we conducted field testing with an ambulance and fire truck in Stockholm, Sweden. During testing, we observed 80% accuracy of detecting siren sounds within 20 meters. When the emergency vehicle is at a longer distance than 20 meters or enables the siren for a very short period of time, the model achieves low accuracy.

Model and hardware specifications

Model specifications

The DEEPCRAFT™ Ready Model for Siren Detection consist of several convolutional neural networks[1] that capture the information from the input data features. The input data are the Fourier transformation[2] of the audio waveform within a window of 500 milliseconds. In particular, we collect all the audio samples for 500 milliseconds, extract the Fourier transformation features, and provide the features to the model to predict the presence of a siren. The maximum inference time of the model is 20 milliseconds.

Dataset

Siren Data

Siren sounds have been selected to cover a range of standard emergency vehicle sounds for firetrucks, ambulances, and police vehicles, focusing on but not limited to the US. Including sounds from different sirens, different background noises, and different distances.

There are three primary siren sounds we've focused on that seem to capture the vehicles of interest. As shown in Figure 1 Spectrogram plots for the three main frequency domains, the three siren sounds can be characterized in the frequency domain as follows:

- Slow Sine pitch alternation
- Square-formed pitch alternation
- Fast Sine pitch alternation

[1] https://en.wikipedia.org/wiki/Convolutional_neural_network
[2] https://en.wikipedia.org/wiki/Fourier_transform

Examples from a spectrogram plot:

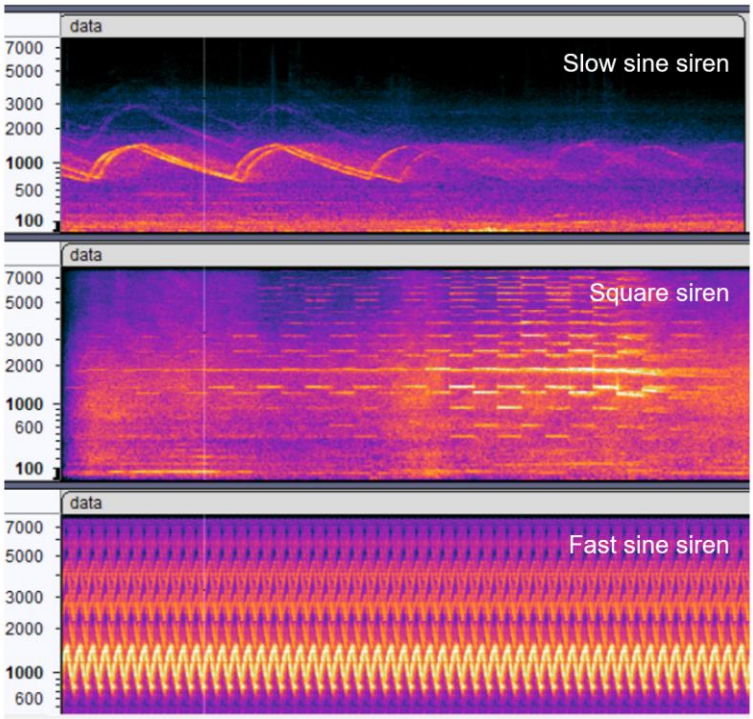


Figure 1 Spectrogram plots for the three main frequency domains

Non-siren Data

The model has also been trained and tested on various non-siren indoor and outdoor sounds. Special care has been taken to:

- Not trigger false positives on sounds where it would be annoying if the use case would be to shut down noise canceling, such as flight cabins, public transport, various machinery like drills and saws, etc.
- Focus on sounds, other than sirens, that also have alternate or move across pitch, such as string trimmer, circle saw, electric shaver, cate meowing, etc.

Below is a list of sounds included, though some categories, like “public transport,” are broad and contain a bit of everything. Also, the siren sounds generally contain background sounds from their natural environment that are not listed below.

- Alarm bell
- Birds
- Car horns
- Chainsaw
- Church bells
- Crowd
- Dog
- Drill
- Electric toothbrush
- Firework
- Footsteps
- Kitchen machines
- Leaf blower
- Music
- Office
- Public transport
- Scooter engine
- String trimmer
- Traffic noise
- Tv and radio broadcast
- Various alarms bells
- Water dripping
- Wind
- Bells
- Blender
- Cat
- Children playing
- Circular saw
- Dinner
- Doorbell
- Electric shaver
- Elevator tone
- Flight takeoff and landing
- Keyboard typing
- Laughter
- Melodies blended with background sounds like traffic.
- Ocean waves
- Phone alarms
- Rain
- Speech
- Thunderstorm
- Train passing
- Vacuum cleaner
- Vehicle engine
- Whistling
- Wood chopping

Hardware Specifications

The model has been prepared for being deployed on an Infineon PSOC™ 6. The minimum hardware requirements are the following:

Memory footprint:

- RAM: 60 kB
- Flash: 51 kB

IoT sense expansion kit:

- (CY8CKIT-028-SENSE).

More details about the hardware setup used, and how to test our model can be found in Appendix 3.

Model performance

We evaluated our model on realistic datasets and field testing.

Real dataset evaluation

For the real dataset evaluation, we extract the confusion matrix from the DEEPCRAFT™ Studio on sounds not used during the model training. Sirens were divided into more prominent sirens that were louder and closer vs less prominent ones that were further away and had lower volume. The criteria are that the siren must go on for at least a few seconds at a higher RMS value than -20. These more prominent sirens represent over 60% of the data.

Time series level confusion matrix

Figure 2 illustrates the confusion matrix on the time series level analysis. This means that we evaluate the model performance in a time-series way, where the audio is processed in a streaming fashion and analyzed by the model. When a positive siren input is inserted in the model, then the model should respond with a positive value, and vice versa. This confusion matrix is based on the output of the model every 500ms. We observe that the model correctly caught 93% of these timesteps as true negatives among the non-siren data. For the sirens, it was able to catch around 78%. It is worth noting that a share of these sirens is distant, so we would not expect the model to be able to distinguish them all too well, and for the use case, it is enough to capture them when they become a bit closer, and a resolution of 500 ms is not needed.

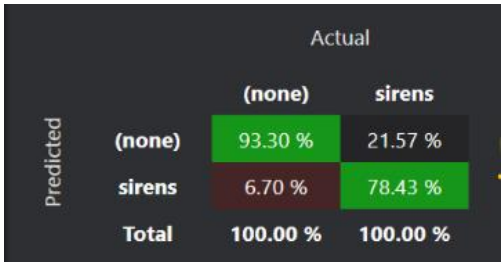


Figure 2 Time series level confusion matrix

File-level confusion matrix

Figure 3 presents the confusion matrix on the same data on the file level. The main difference between time series and file-level evaluation is that we focus on evaluating the model performance by detecting the presence of at least one positive siren example in the file. We observe that the model triggers a false prediction event on 2.6 % of the non-siren files while capturing a bit more than 80% of the sirens that still contain siren files that are distant and/or short.

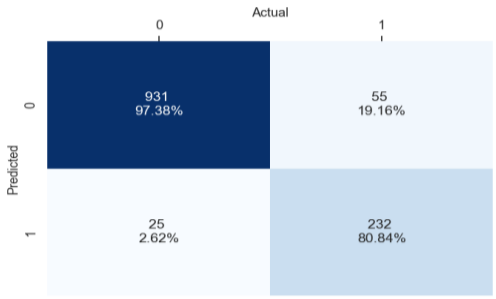


Figure 3 File-level confusion matrix

Corner cases

Figure 4 illustrates the percentages of triggers on file level per category. The plot shows the model can effectively capture closer sirens while suppressing triggers on various non-siren background sounds. This is with the default post-processing, but the customer can adjust the model's sensitivity depending on preference on capturing more positives or suppressing more negatives.

The most difficult false positives to avoid for the model are generally various *alarms*. While we significantly suppressed them, they are *not necessarily harmful* to become aware of, such as in the use case of shutting off noise canceling. Whistling also showed some false positives, but it is not assumed to be a significant part of someone's everyday life.

For more detailed results in all categories, please review Appendix 1.

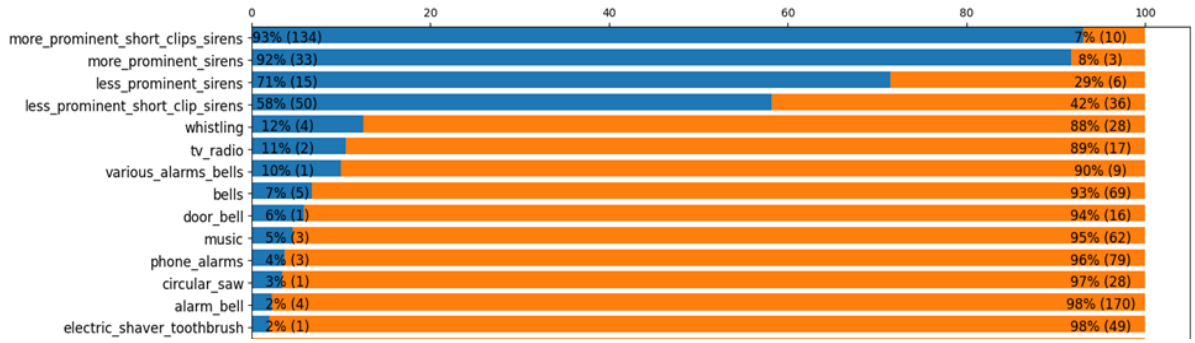


Figure 4 Histogram of model predicting that the file contains siren vs not containing siren

In Figure 5 we present the false positives that triggered the model for non-siren sounds. We observe, that if someone was constantly whistling, it would for example trigger roughly more than once per minute as a false positive. For categories like music, television and radio, these sounds are likely to be more in the foreground of the audio which is not necessarily the case in noisy environments where it's blended with other sounds and less likely to trigger a response from the model.

There are some categories like circle saw, which might require the ability to toggle off the feature if suppressing noise cancellation on a job where this is used, but most likely, one would rather wear robust hearing protection than headphones when systematically working with things like that.

For more detailed results in all categories, please review Appendix 1.

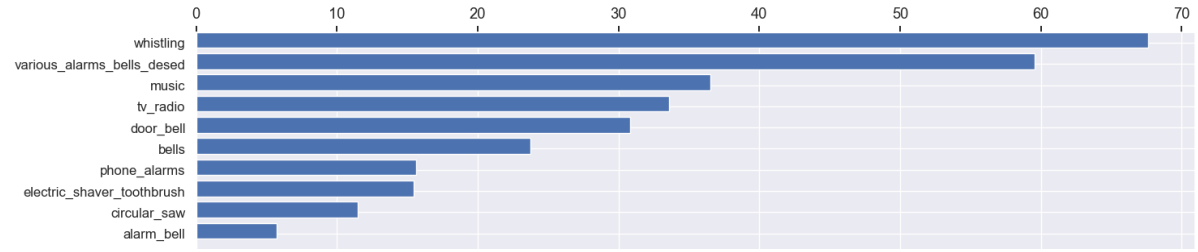


Figure 5 False positive siren event triggers per hour on non-siren sounds.

Field testing

Ambulance testing on device *in Western Stockholm*

We conducted the ambulance field testing at an ambulance station in the western parts of Stockholm. In this experiment, both the tester and the ambulance were static, and the ambulance driver was triggering the siren alarm to monitor the model reaction. We tested both the Square form and fast Sine form siren. Based on our experimentation, the model satisfies the 50 m requirement we've had as a target.

Ambulance drive by

In this set of experiments, the tester was static, and the ambulance drove and passed by the tester. The purpose of this experiment is to monitor the model performance in a more realistic scenario than before, where the emergency vehicle is approaching the pedestrian. In this test, the ambulance driver had the siren activated, and it drove by the tester on the street three times. We could observe that the model did indeed trigger every time, and the model was accurate even at a distance notably longer than 50 meters, which is our target.

Ambulance static and tester walking

In this set of experiments, the tester started close to the ambulance and then walked away. During these tests, we could observe that the model constantly triggers predictions at pretty high probabilities.

Fire Truck Testing in Stockholm

Similarly, we conducted experiments on a firetruck. In this set of experiments, we observed similar results as the ambulance. It is worth noting, that when the firetruck was driving further away from the tester, in distances more than 50 meters, then the model accuracy was deteriorating.

More details about the field testing can be found in Appendix 2.

Appendix 1

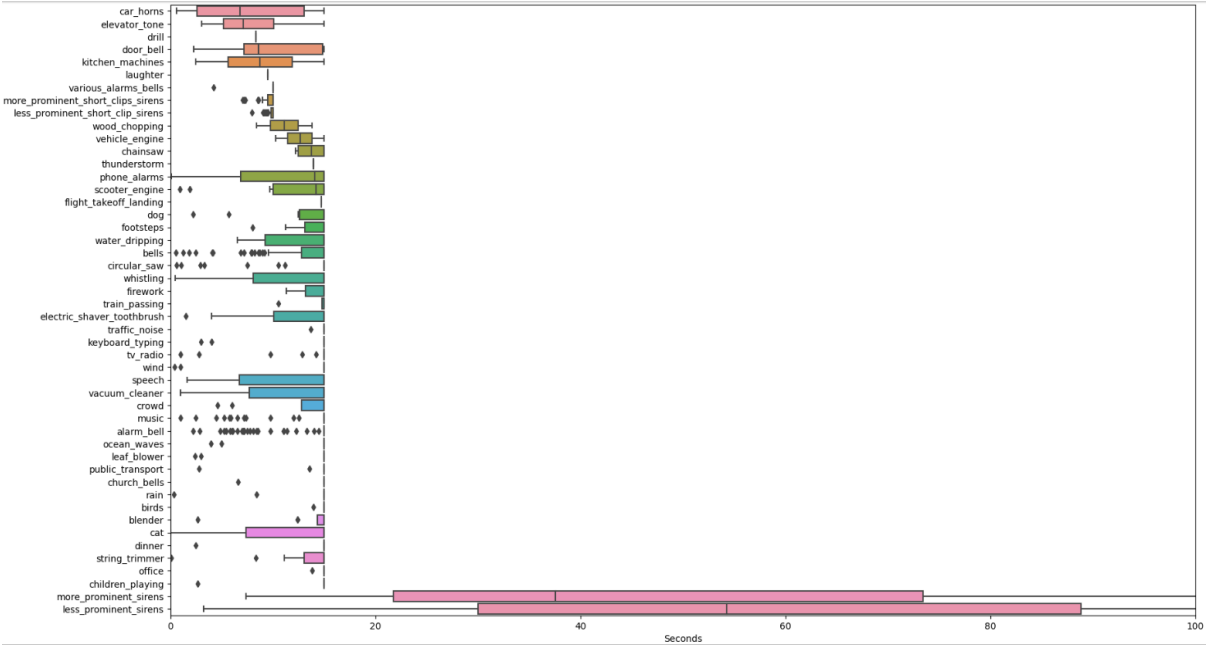


Figure 6 Distribution of length of files across categories in validation set

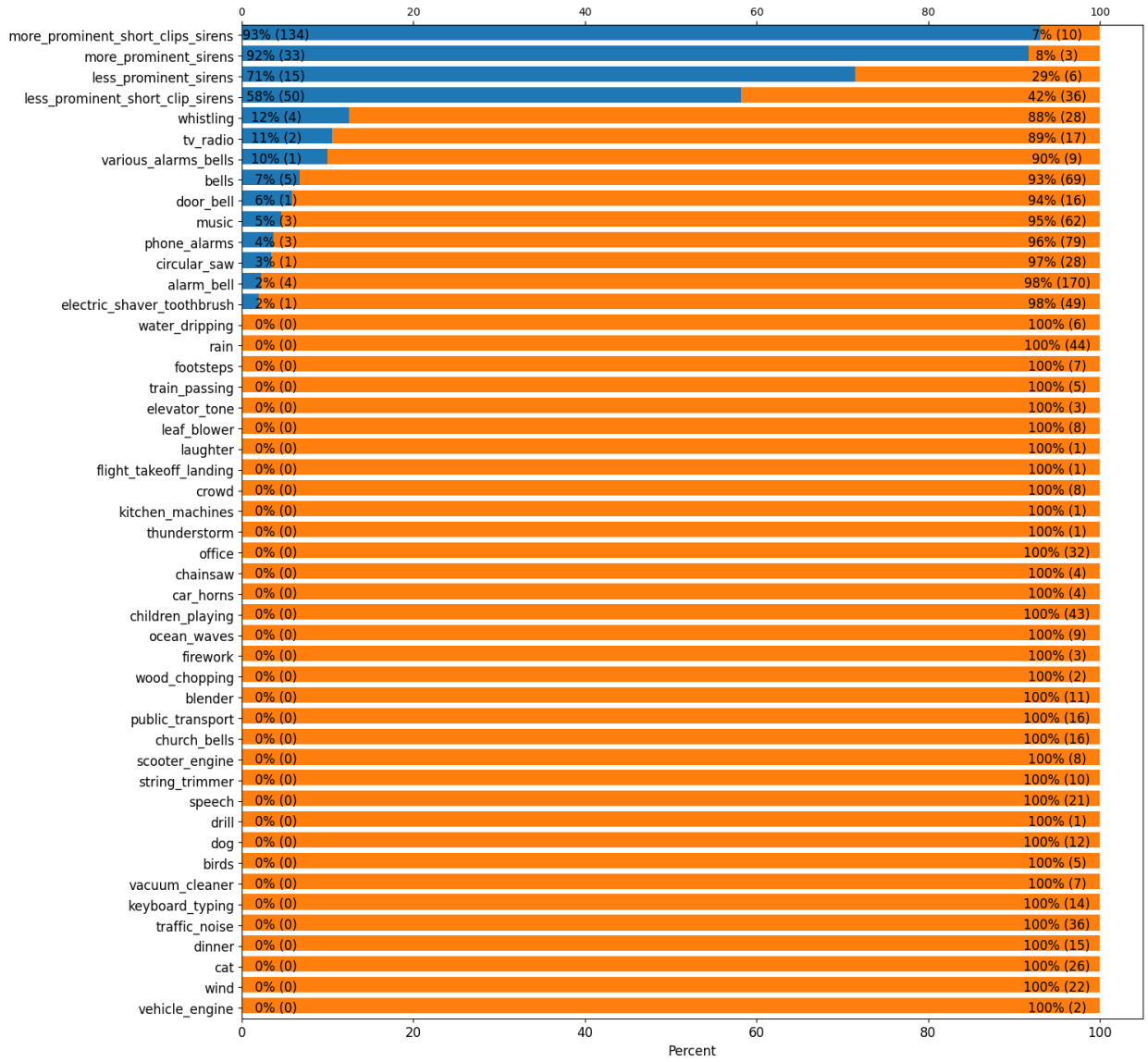


Figure 7 File level histogram of model predicting that the file contains siren vs not containing siren

Notes for this plot:

- The percentages mean how many triggers there are on file level (given the post processing)
- The values in parenthesis after the percentages on the bars show how many files the percentage represents
- The non-siren background sounds for the validation set have been split into 15 sec file segments for more fair comparison across categories

The plot shows that the model can effectively capture closer sirens while suppressing triggers on a wide range of the non-siren background sounds. This is with the default post processing but the customer can adjust the sensitivity of

the model depending on preference on capturing more positives or suppressing more negatives.

The hardest false positives to avoid for the model are generally various *alarms*, and while we significantly suppressed them, they are *not necessarily harmful* to become aware of, such as in the use case of shutting off noise canceling. Also, whistling has been showing some false positives, but it is not assumed to be a major part of someone's everyday life.

False positive triggers per hour on non-siren background data

Another way to evaluate the results is to simulate how many false positives the model would trigger on the non-siren background categories per hour. We do this by:

- Dividing this background data into 5 second chunks, as a proxy for counting “annoyances” for a user being nudged by a false positive
- Evaluating if the model would have triggered a siren prediction for each chunk given the default post processing (only counting maximum one trigger per 5 second window)
- Calculating the ratio of total number of triggered siren events / hour

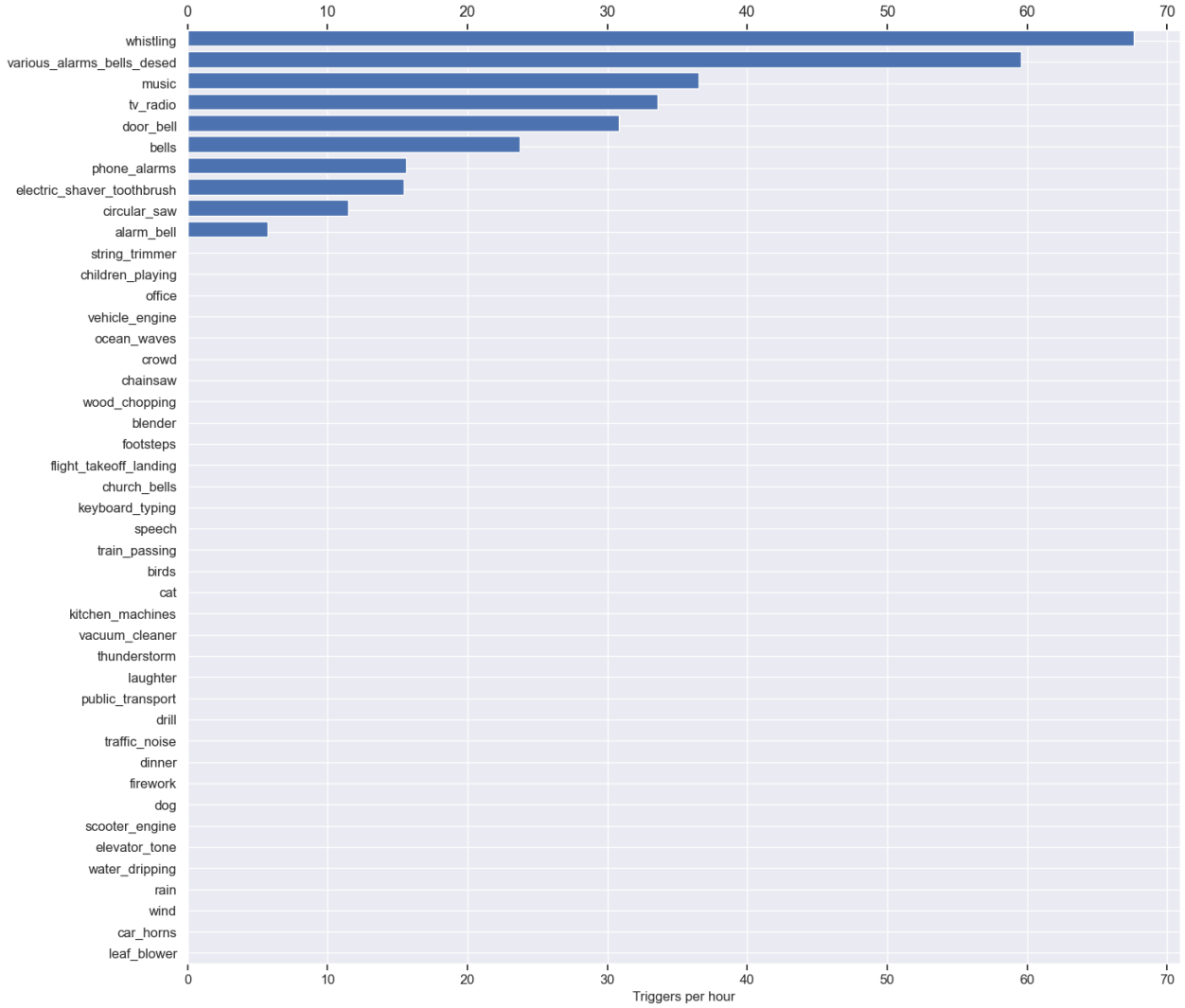


Figure 8 False positive siren event triggers per hour on non-siren sounds

What we can see here is that if someone was constantly whistling, it would for example trigger roughly more than once per minute as a false positive. For categories like music, television and radio, these sounds are likely to be more in the *foreground* of the audio which is not necessarily the case in noisy environments where it's blended with other sounds and less likely to trigger a response from the model.

There are some categories like circle saw which might require the ability to toggle off the feature if suppressing noise cancellation on a job where this is used, but most likely one would rather wear robust hearing protection than headphones when systematically working with things like that.

Note once again that we *effectively suppress several sounds completely* from triggering in the validation set.

Average predicted siren probability per non-siren category

Lastly, another way to evaluate the results is to look at what the model predicts on average across all time steps for each category. This is sometimes also called the confidence of the model that a data point belongs to a certain class (here: Sirens).

Stated differently, this means what the model thinks the probability of a given data point being a siren is. For non-siren classes we want these probabilities to be as low as possible showing that the model is not reacting to them.

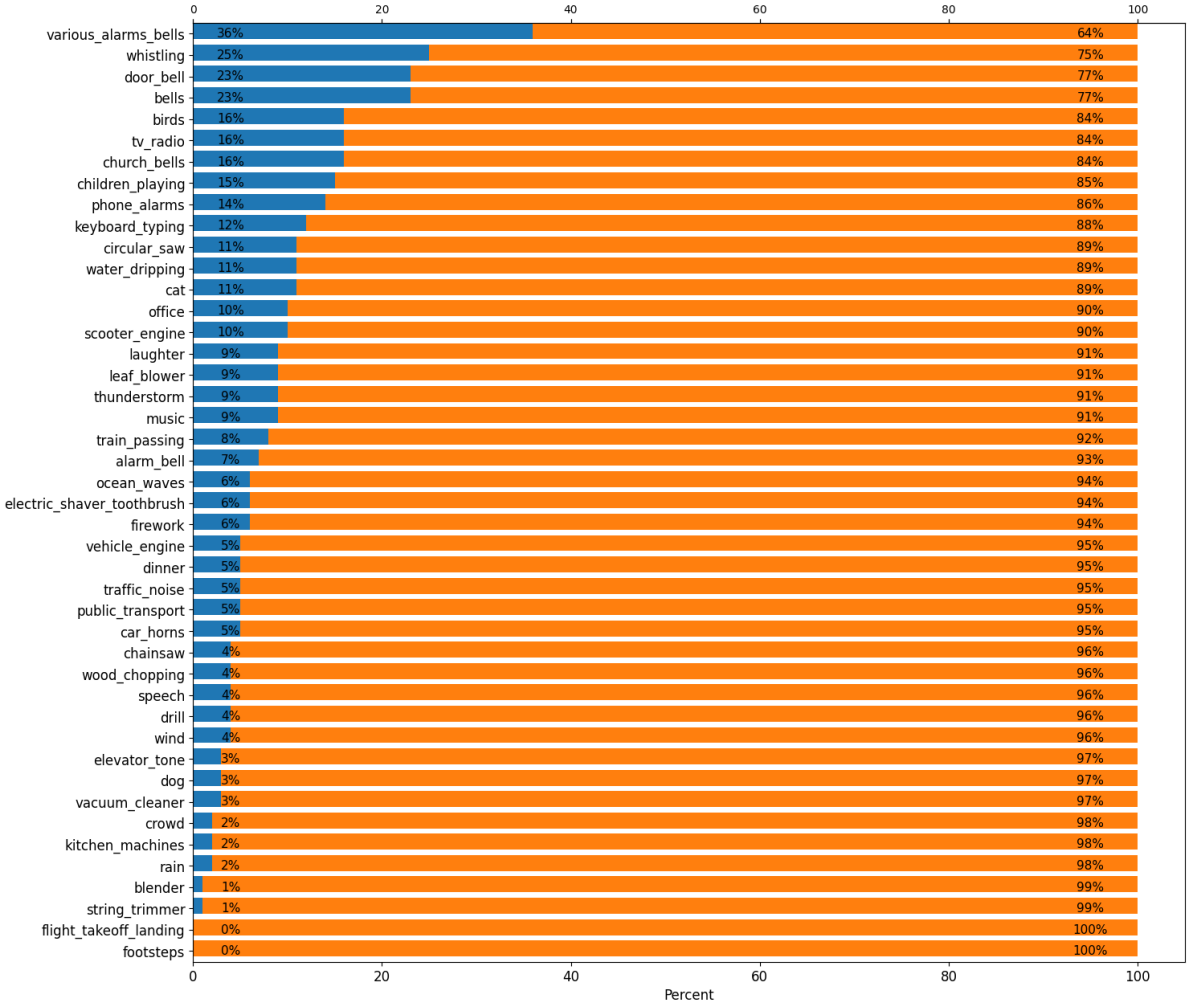


Figure 9 Average predicted siren probability for sounds across samples per category

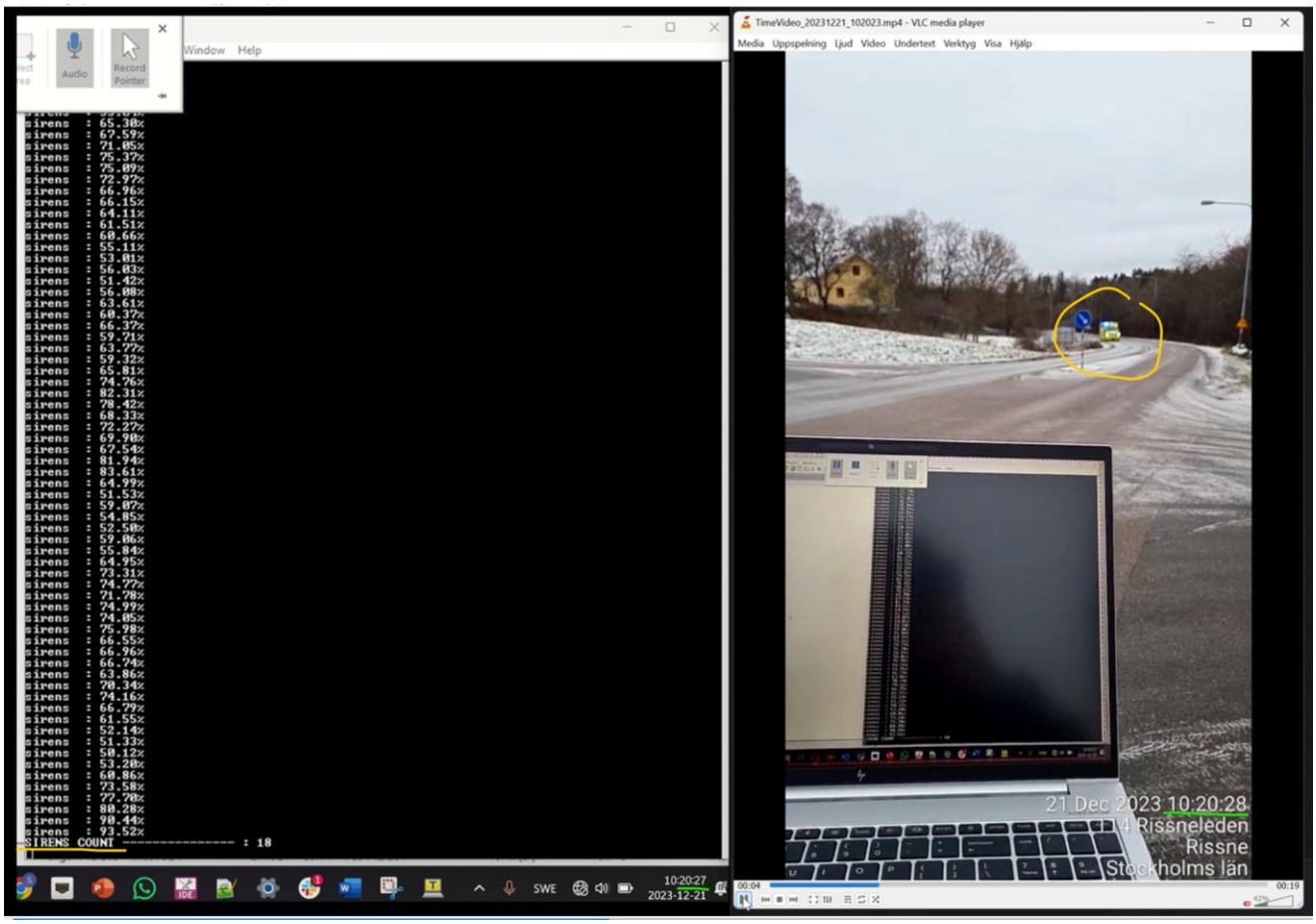
We can see here that the model is averaging really low for several sounds which supports that we don't need to be too concerned about false positives for those (like drills, speech, kitchen machines, flight takeoff and landing etc.).

Other considerations for false positives

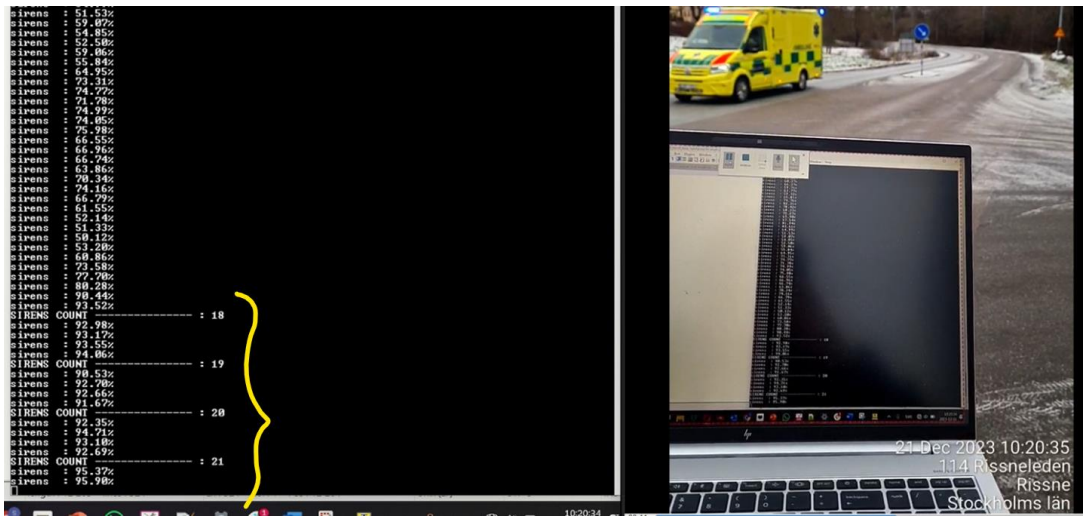
An even more effective way of suppressing false positives for other contexts than when a pedestrian is out and walking down a street, would be if the model could be toggled on/off based on some intelligent feature such as GPS location change. Then it could be switched off from triggering any false positives if staying at home or at work, and come online again once the person is walking down a street.

Appendix 2

Model triggering siren event when siren approaches from the distance



Model continuing to trigger siren events as ambulance approaches



We did however note that the model, in one of the three drives, using the post processing filtering, missed some seconds when the vehicle was getting closer and swapping to the Fast Sine wave form (which is usually not a problem). It did however catch the siren initially. In the use case of fading out music and removing noise cancelling in headphones, it would still have worked well if the feature had a cooldown period of a few seconds, and the user would still have become attentive based on the two initial event triggers when the ambulance approaches in the far end of the curve in the picture.

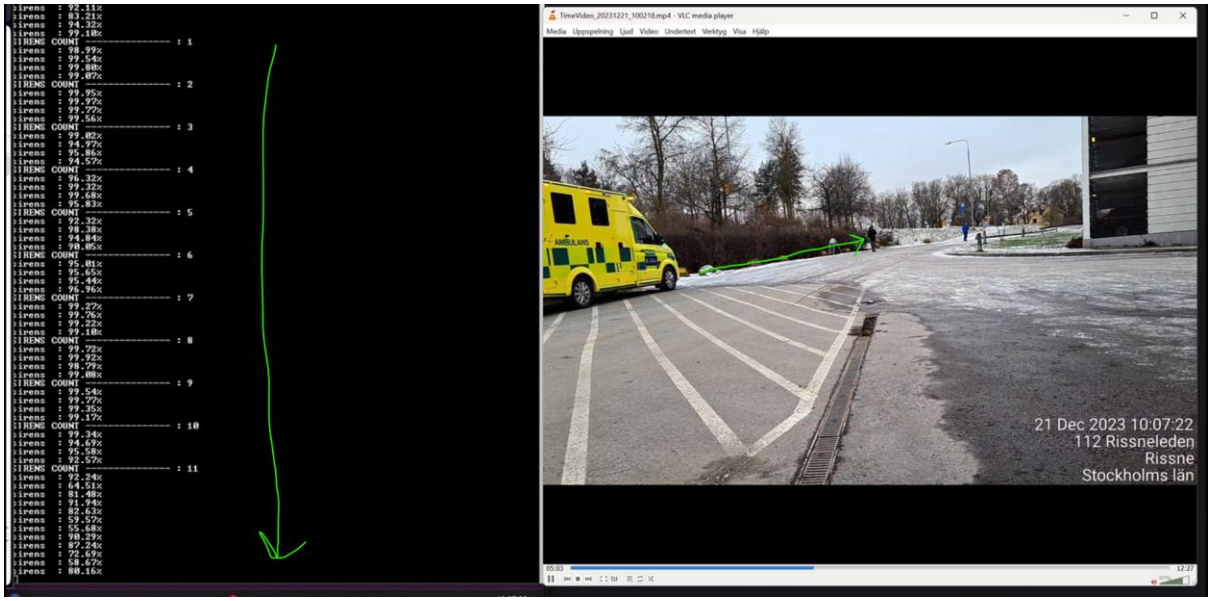
Ambulance standstill and varying distance by walking

In these tests we started close to the ambulance and then walked away. We can see that the model more or less constantly triggers predictions at pretty high probabilities.

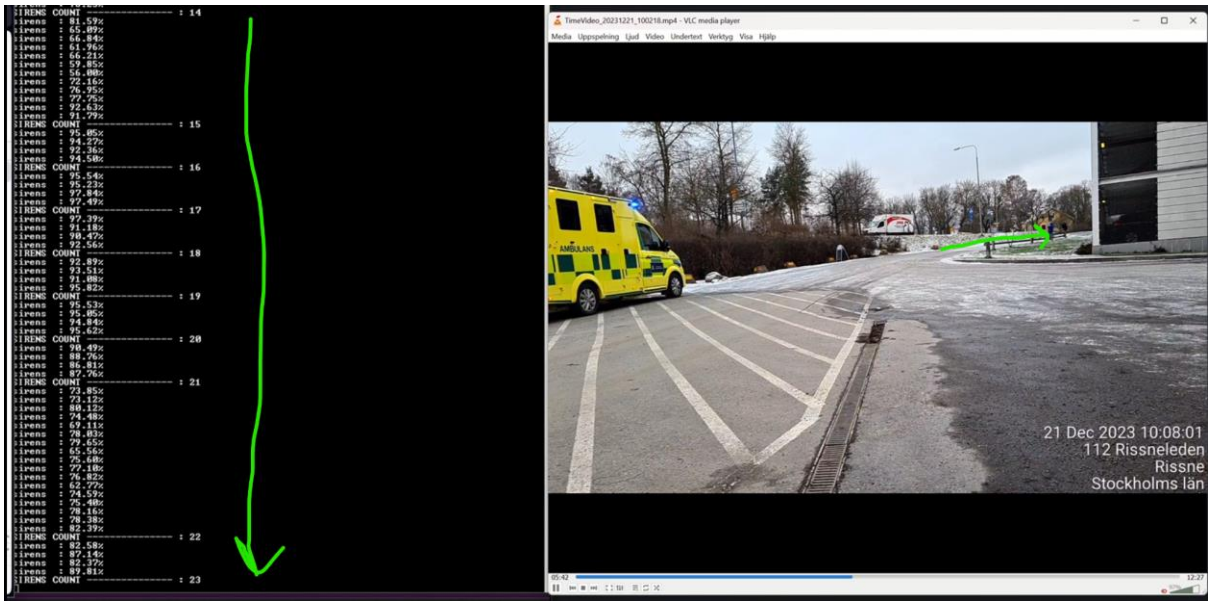
We do note once again that it can happen for a few seconds that the predictions don't get past the post processing policy and therefore would not trigger the feature for those time steps.

Square form siren

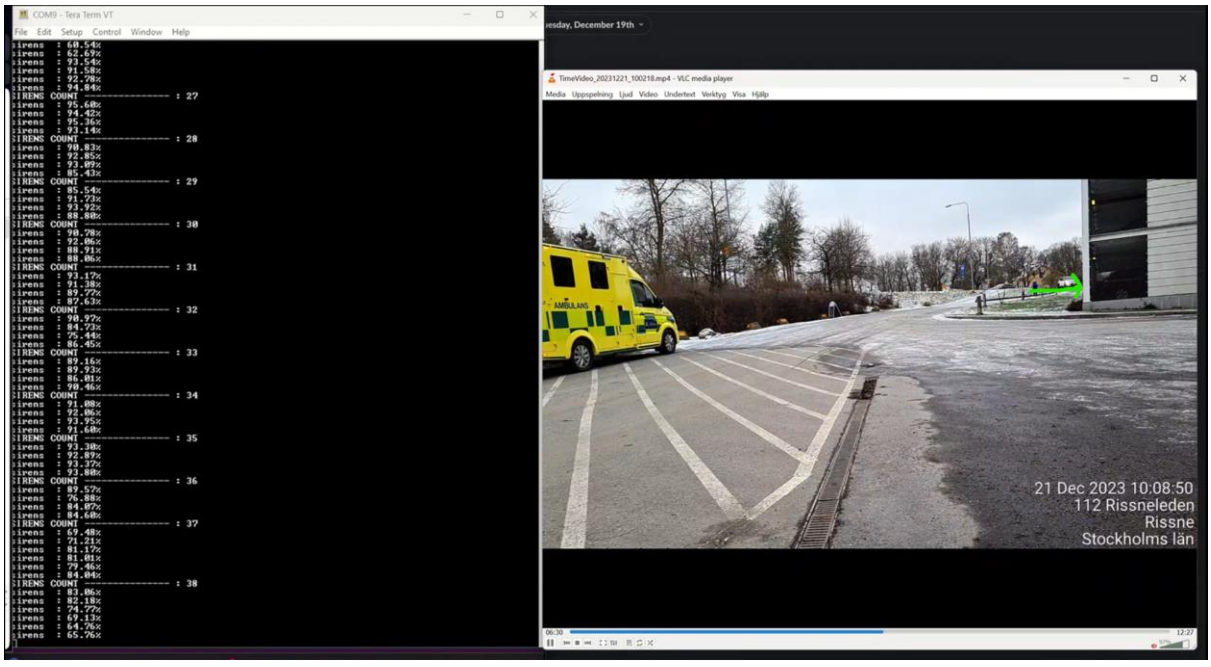
In the first screenshot walking away from the vehicle, the siren event is triggers around 80% of each occasion (11 / 11 + 3) where the post-processing is successfully passed, with a gap in the end.



In the next picture, approaching around 80 m distance from the vehicle, the model captures around ⅔ of the events (10 / 10 +5).



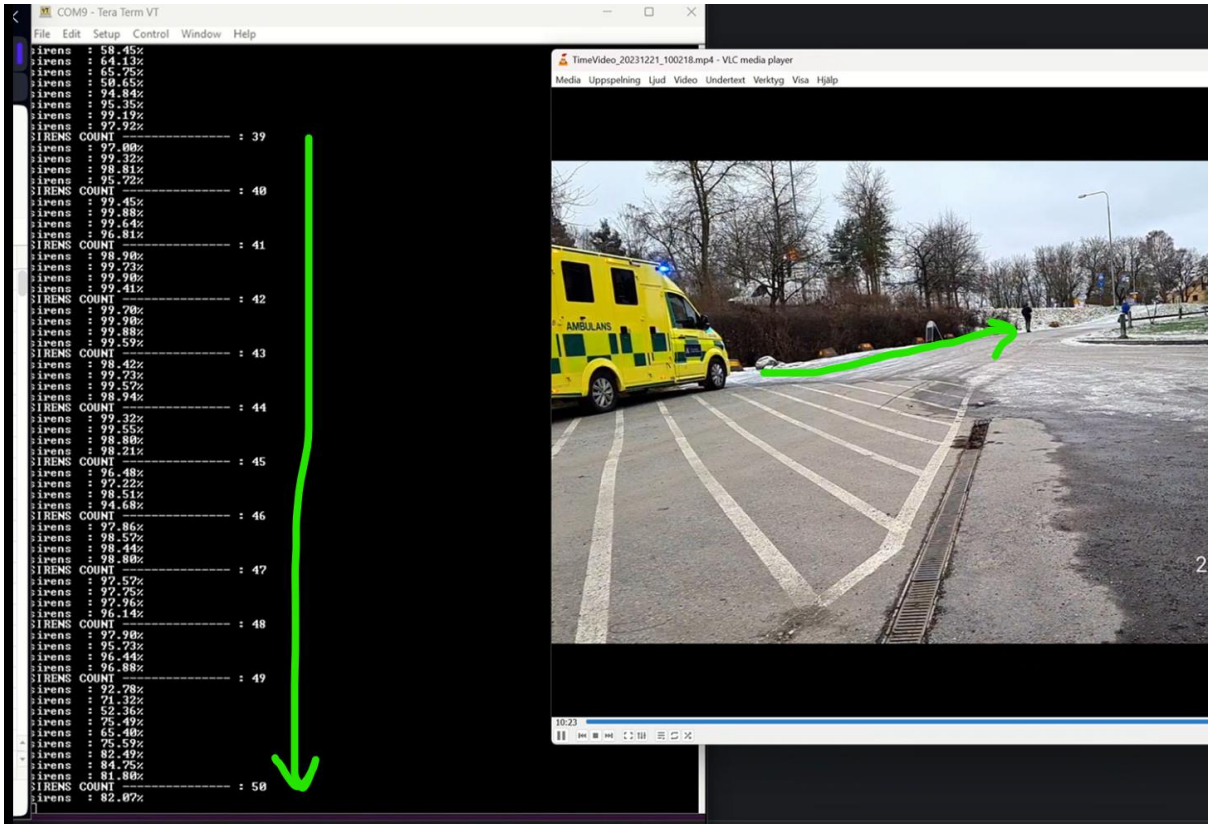
Lastly, when standing at around 90 - 100 m distance (a bit further behind the building in the picture), the model overall did reasonably well. It did have some gaps but overall triggering around 75% of the time past the post processing, which supports that we can capture vehicles in time as they approach from distance.



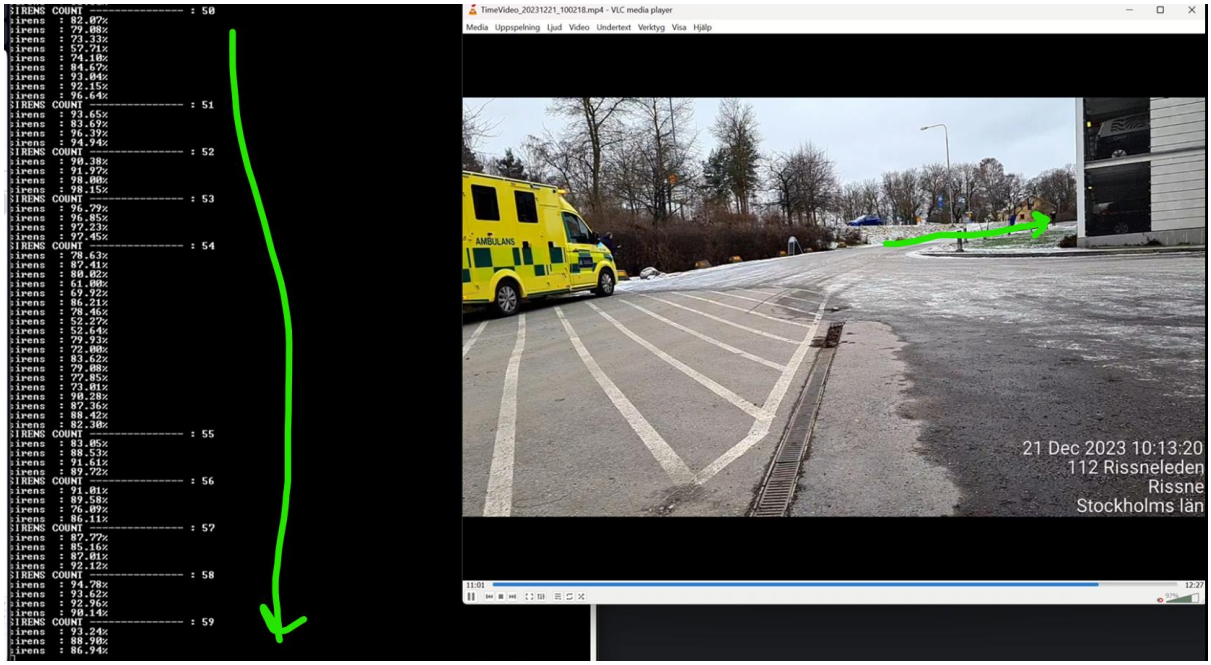
Square form mixed with Fast Sine siren

This test is similar to the previous test but mixing the Square formed sirens with the Fast Sine sirens.

From the first walking distance we can see that despite alternating so there was 2 Fast Sine and 2 Square siren sounds, the model only missed passing one step through the post processing (92% recall).



For this next section which, should be noted, is a longer distance, the model had one gap that was a bit longer, over several seconds, containing mostly the Fast Sine siren. The model did however infer above 70% probability for many of these timesteps, and caught 4+ consecutive events passing the post processing both before and after this gap.



Final remarks on static ambulance tests

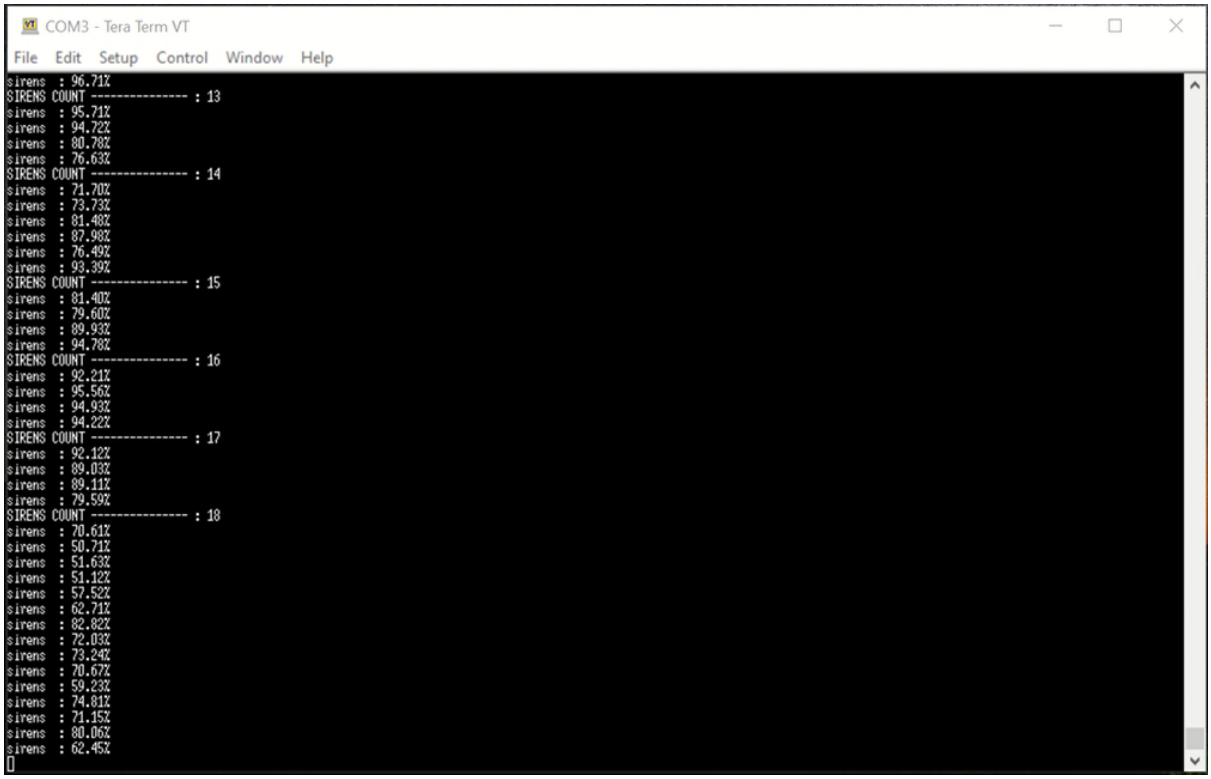
One thing to note is that the natural state of emergency vehicles is to be in a hurry, not standing still for many seconds. It could be that the model finds it easier to identify the real approaching siren because it has learnt that there should in most cases be some slight Doppler effect as it is common that the vehicles are on the move.

Square form siren

In the first screenshot walking away from the vehicle, the siren event is triggers around 80% of each occasion (11 / 11 + 3) where the post-processing is successfully passed, with a gap in the end.

Fire Truck Testing in Stockholm

Similarly testing was performed on a firetruck. Here it can be seen that when the firetruck was close by multiple detections were triggered. 18 in total for the same firetruck but as it went further away the model was unable to successfully detect it.


COM3 - Tera Term VT
File Edit Setup Control Window Help
sirens : 96.71%
SIRENS COUNT ---------- : 13
sirens : 95.71%
sirens : 94.72%
sirens : 80.78%
sirens : 76.63%
SIRENS COUNT ---------- : 14
sirens : 71.70%
sirens : 73.73%
sirens : 81.48%
sirens : 87.98%
sirens : 76.49%
sirens : 93.39%
SIRENS COUNT ---------- : 15
sirens : 81.40%
sirens : 79.60%
sirens : 89.93%
sirens : 94.78%
SIRENS COUNT ---------- : 16
sirens : 92.21%
sirens : 95.56%
sirens : 94.93%
sirens : 94.22%
SIRENS COUNT ---------- : 17
sirens : 92.12%
sirens : 89.03%
sirens : 89.11%
sirens : 79.59%
SIRENS COUNT ---------- : 18
sirens : 70.61%
sirens : 50.71%
sirens : 51.63%
sirens : 51.12%
sirens : 57.52%
sirens : 62.71%
sirens : 82.82%
sirens : 72.03%
sirens : 73.24%
sirens : 70.67%
sirens : 59.23%
sirens : 74.81%
sirens : 71.15%
sirens : 80.06%
sirens : 62.45%

Appendix 3

Testing

On-device testing

To test the model, proceed in the following steps:

1. Obtain the ready model library from Imagimob
2. Obtain a PSOC™ 6 board. E.g. PSOC™ 62S2 Wi-Fi BT Pioneer Kit with IoT sense expansion kit (CY8CKIT-028-SENSE)
3. Create an example project that samples the microphone. E.g. Deployment on PSOC™ Code Example
4. Use the provided API calls and example code in the library header
5. Create UI for displaying library outputs. E.g. a printf statement to a terminal

Loading model using the static library:

1. Load your target firmware whether this is a code example or target firmware. In the case of code example, we recommend a basic PDM/PCM Audio example
2. Add the provided static gcc library to your project
3. Use the API and code examples provided in the header file
4. Trigger a UI event based on the flag raised by the library. I.e. printf statement that the event occurred

Loading a model using the hex file:

1. Flash the board with the hex file
2. Open a serial terminal to observe the prints. Terminal settings:
    a. COM port is dependent on the computer being used, check device manager to find the port number
    b. Speed: 115200
    c. Data: 8 bit
    d. Parity: none
    e. Stop bits: 1
    f. Flow control: none

Testing the model:

1. Place the device 5-10 metres away from an emergency vehicle
2. Turn on the sound in the vehicle
3. Prediction is generated:
    a. If using custom firmware: check whatever output you have created yourself
    b. If using hex file: check the terminal for prints