# Feature Set Analysis

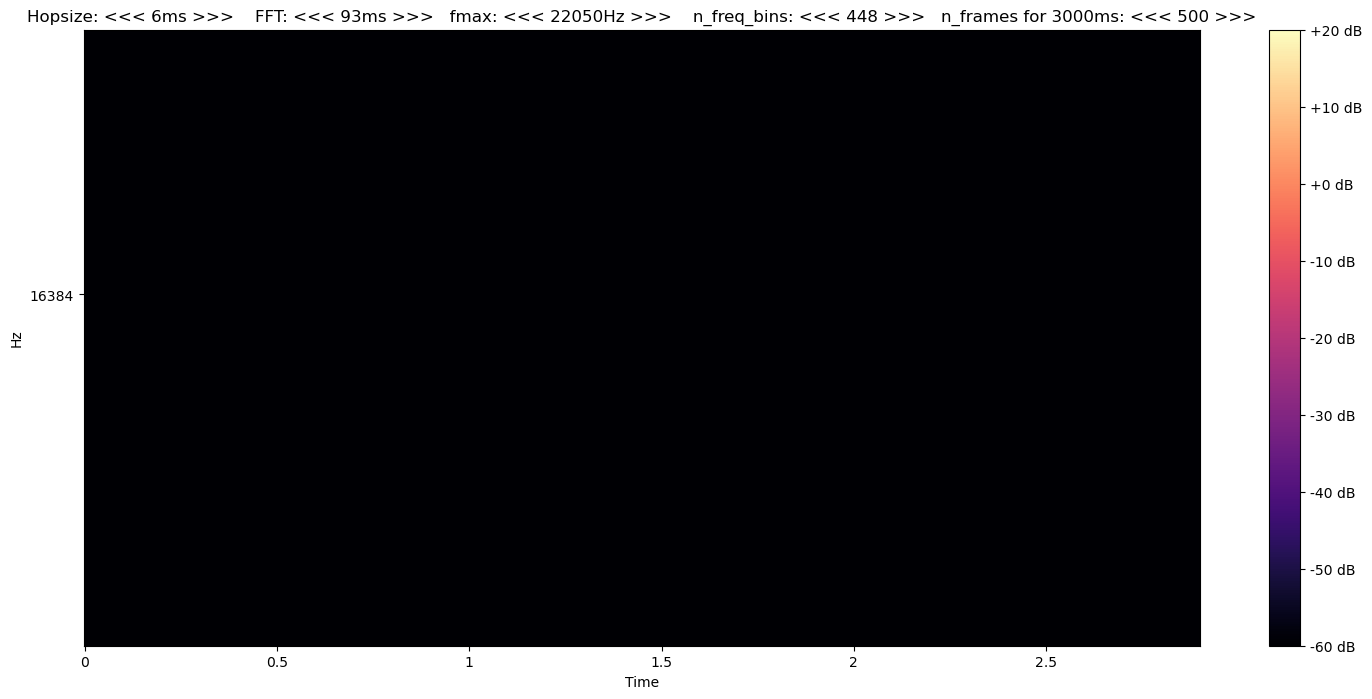
To get a decent performance, first the features that are fed into a machine learning system need to be explored. A spectrogram type was chosen. The Mel spectrogram is the feature set of choice. But even after setting the feature set, there are some parameters to be tweaked to get an optimal performance out of the data:

# Parameters:

Some of them are already fixed and won’t be changed: sample rate(22050Hz), fmin(0Hz), fft\_window\_length\_with\_padded\_zeros(8192 🡪 Hz resolution). The tests and images are from the Jupyter notebook “frequency and time resolution analysis”.

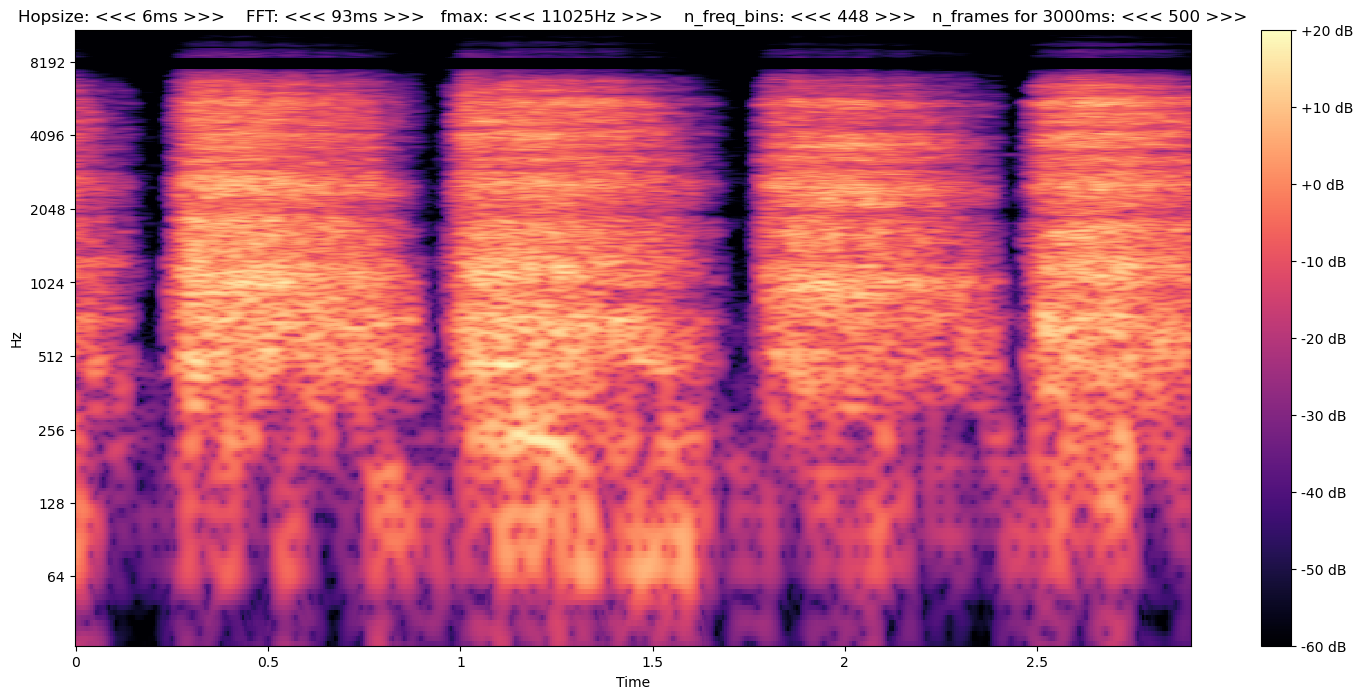
## Fmax:

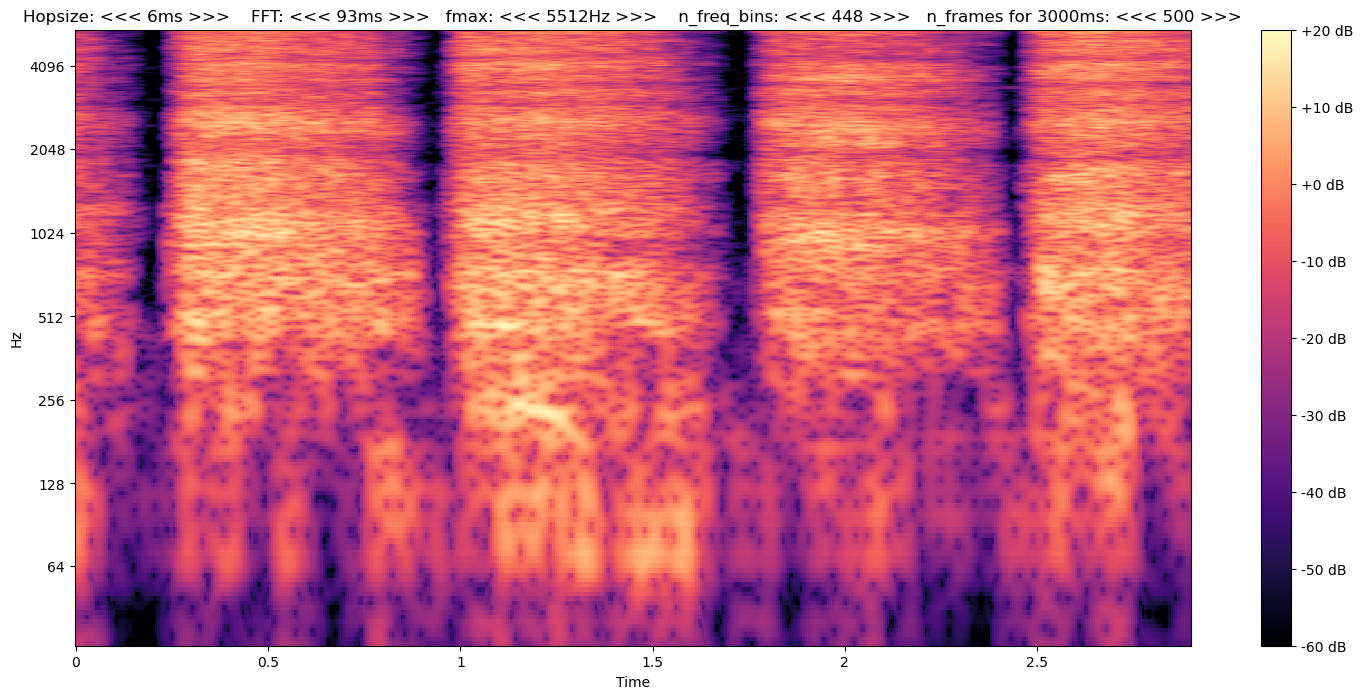
To avoid longer processing time and irrelevant noise in the high frequency regions to cause an instable performance, we want to cut off the spectrogram at a certain frequency. Cut-off Frequencies at fs, fs/2 and fs/4 (22kHz, 11kHz and 5.5kHz) were tested. Above 11kHz there is no signal present above -60dB.

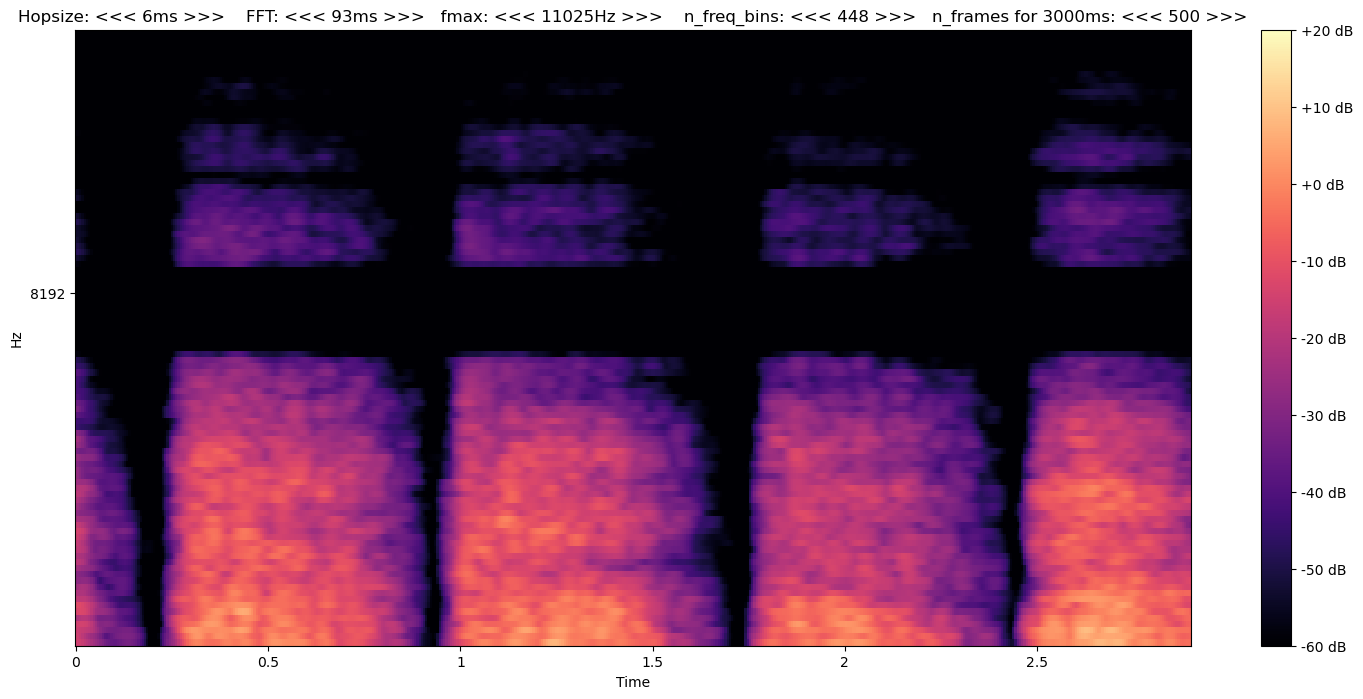


When cutting off at 5.5kHz though, there is quite a bit of signal lost (images: up to 11kHz, up to 5.5kHz, 5.5kHz to 11kHz):

As a result, a fmax of 11kHz was chosen as a good compromise of not cutting of relevant data but keeping the size minimal.

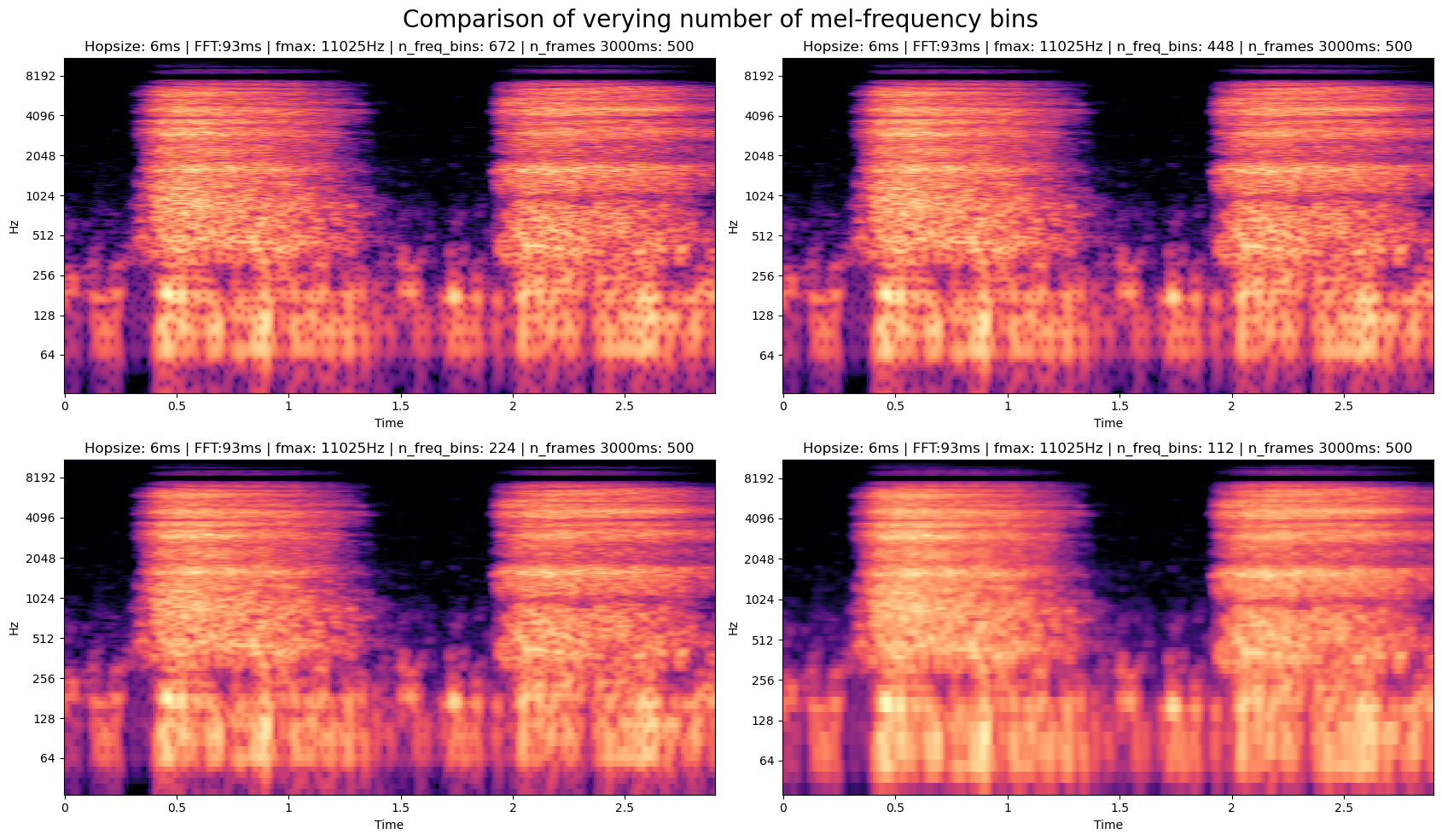


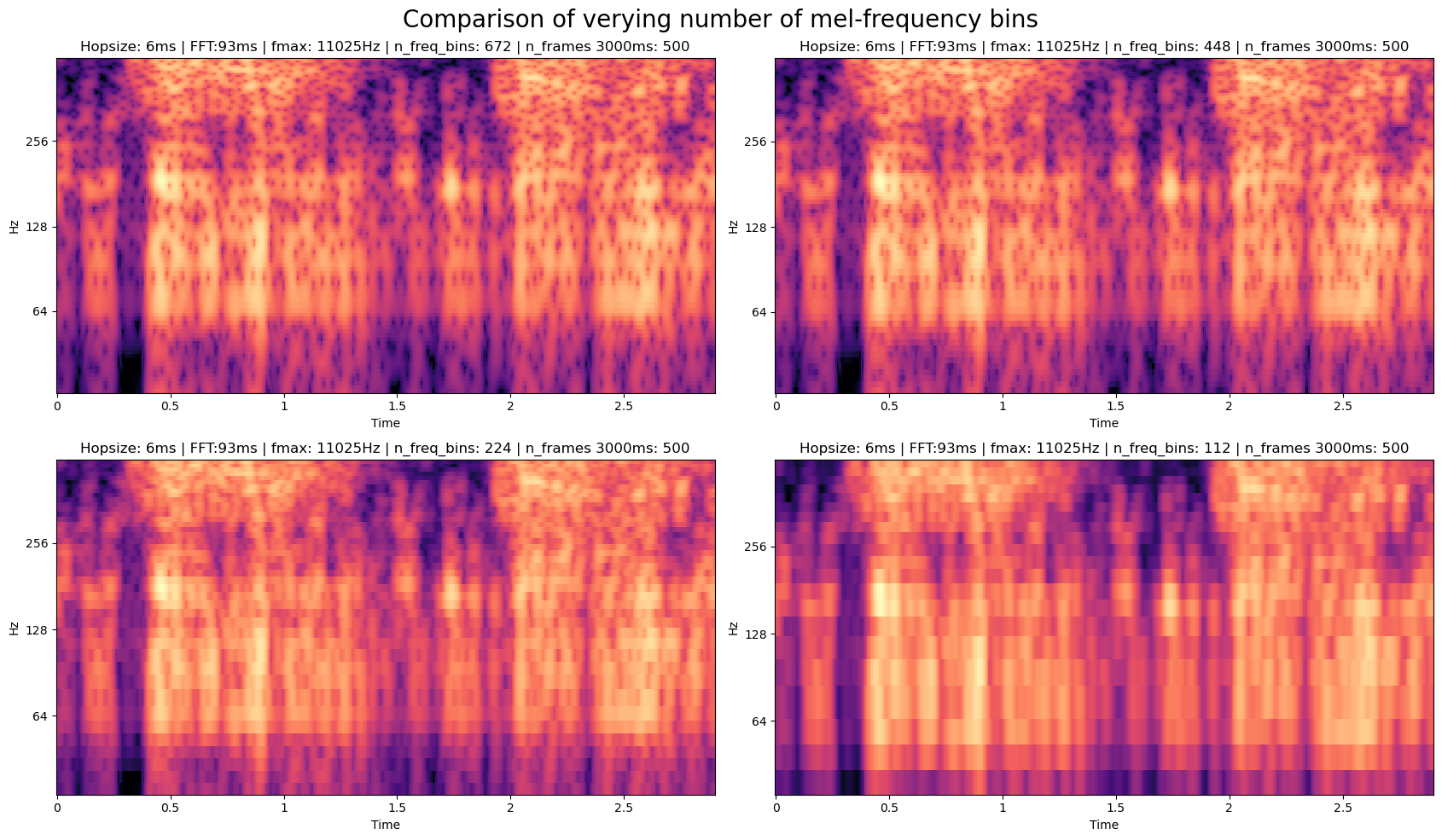


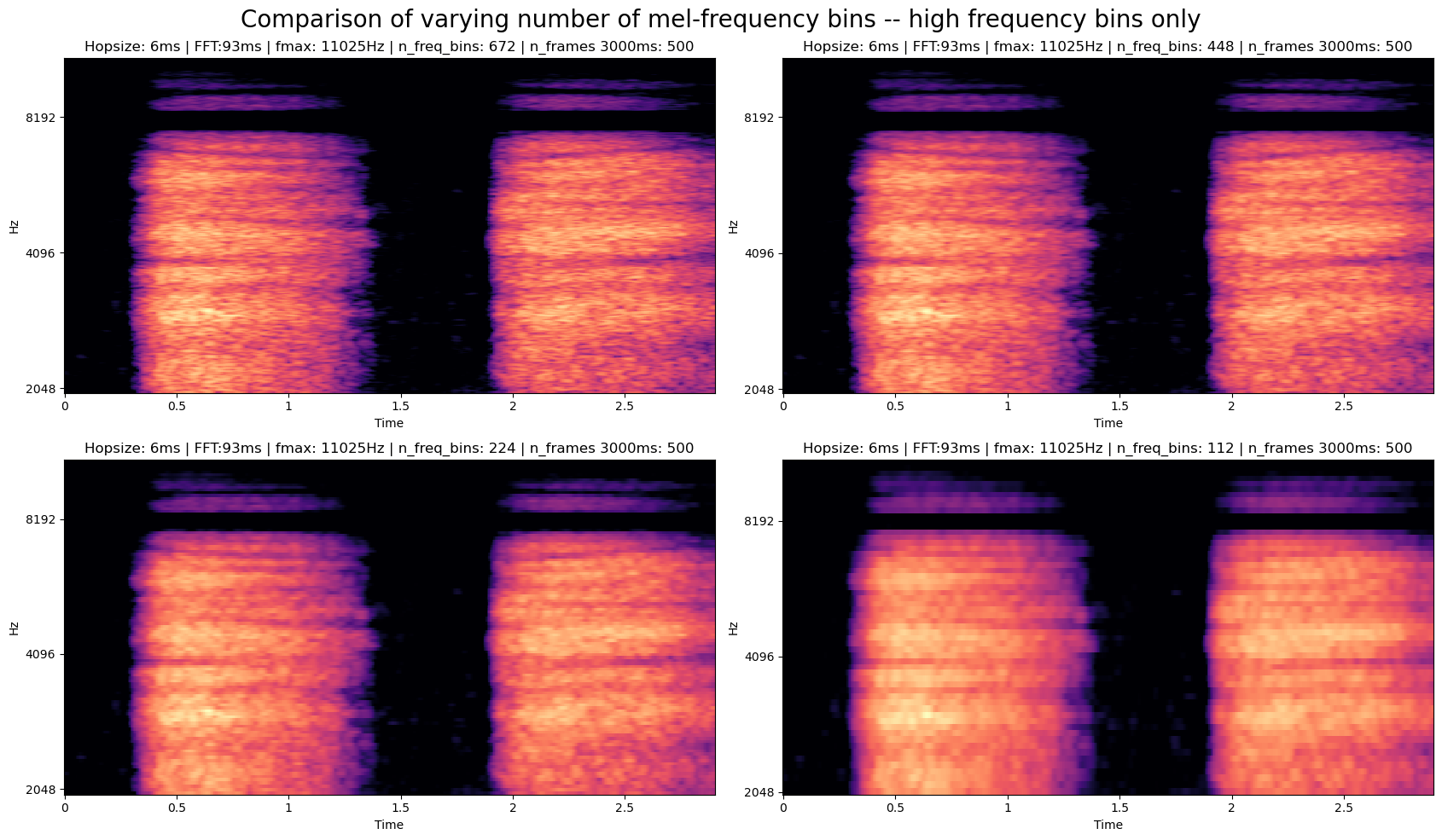


## Frequency Bins (Mel Bins):

The Frequency resolution is the vertical resolution. Especially in the lower frequency regions it might average out some signals over multiple frequencies if the number of frequency/mel bins is not sufficient. Choosing too many bins will maybe result in a lot of noise and variance in the signal. For a visual comparison, we choose a fixed length of **3 seconds, hopsize=6ms, FFT\_window=93ms**







## Results:

It seems that in the low frequencies, there is a very apparent blur when using fewer frequency bins. Though it is more present in the low frequencies, it can also be found in higher frequency regions. 112 bins seems to be too few bins. 224 bins seem to work somewhat but 448 bins gives a decent improvement. Going further up does not really anymore. For the next sections we choose 448 bins as a resolution.

# Hop Size:

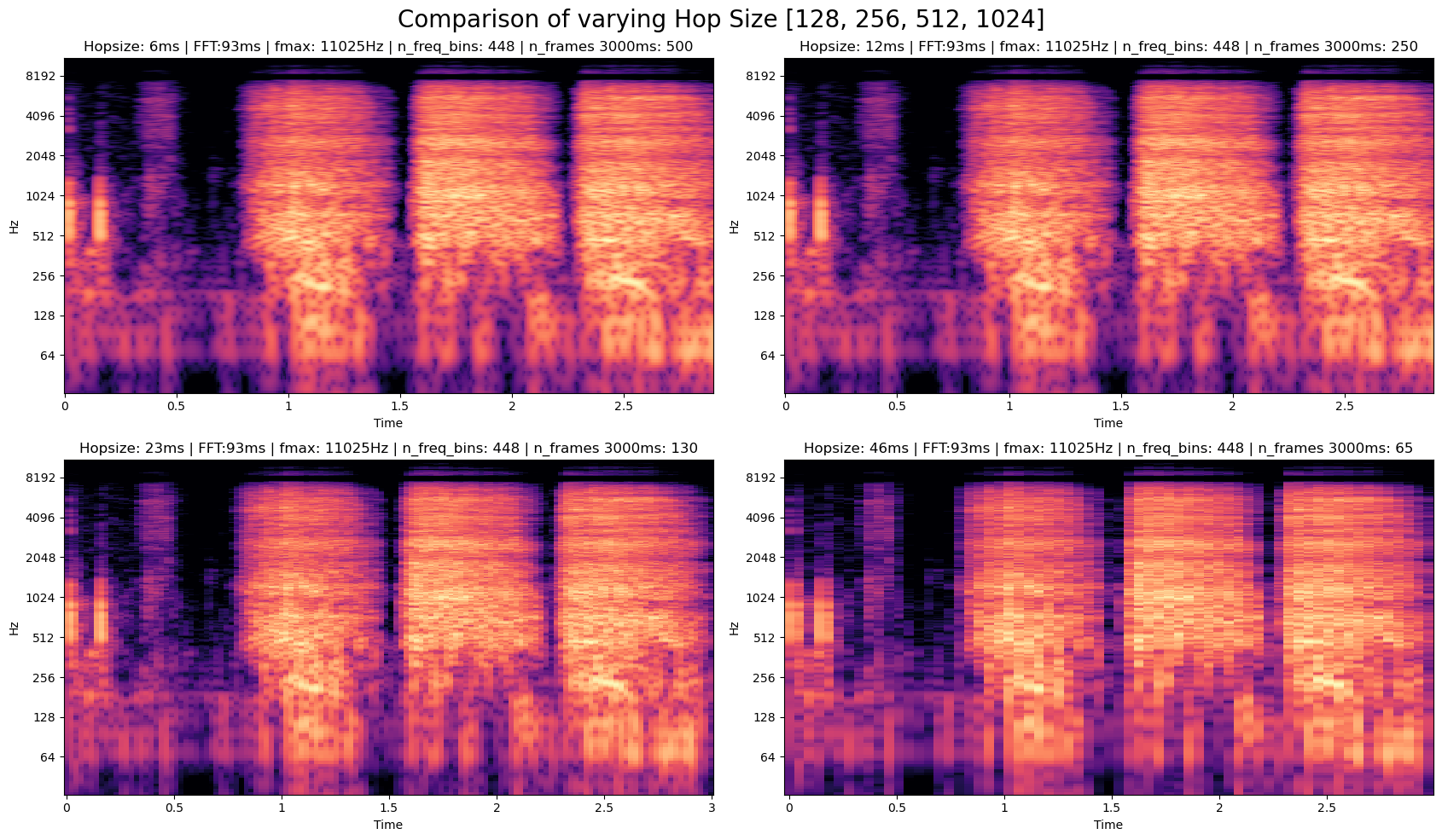
The hop size in combination with the FFT length is responsible with the horizontal resolution/time resolution. The hop size determines how large the total time frame is of the input that is fed into the network. For a fixed number of timeframes, a halving of the hop size doubles the total time in seconds, that the network sees. (We always assume a sampling frequency of 22kHz)

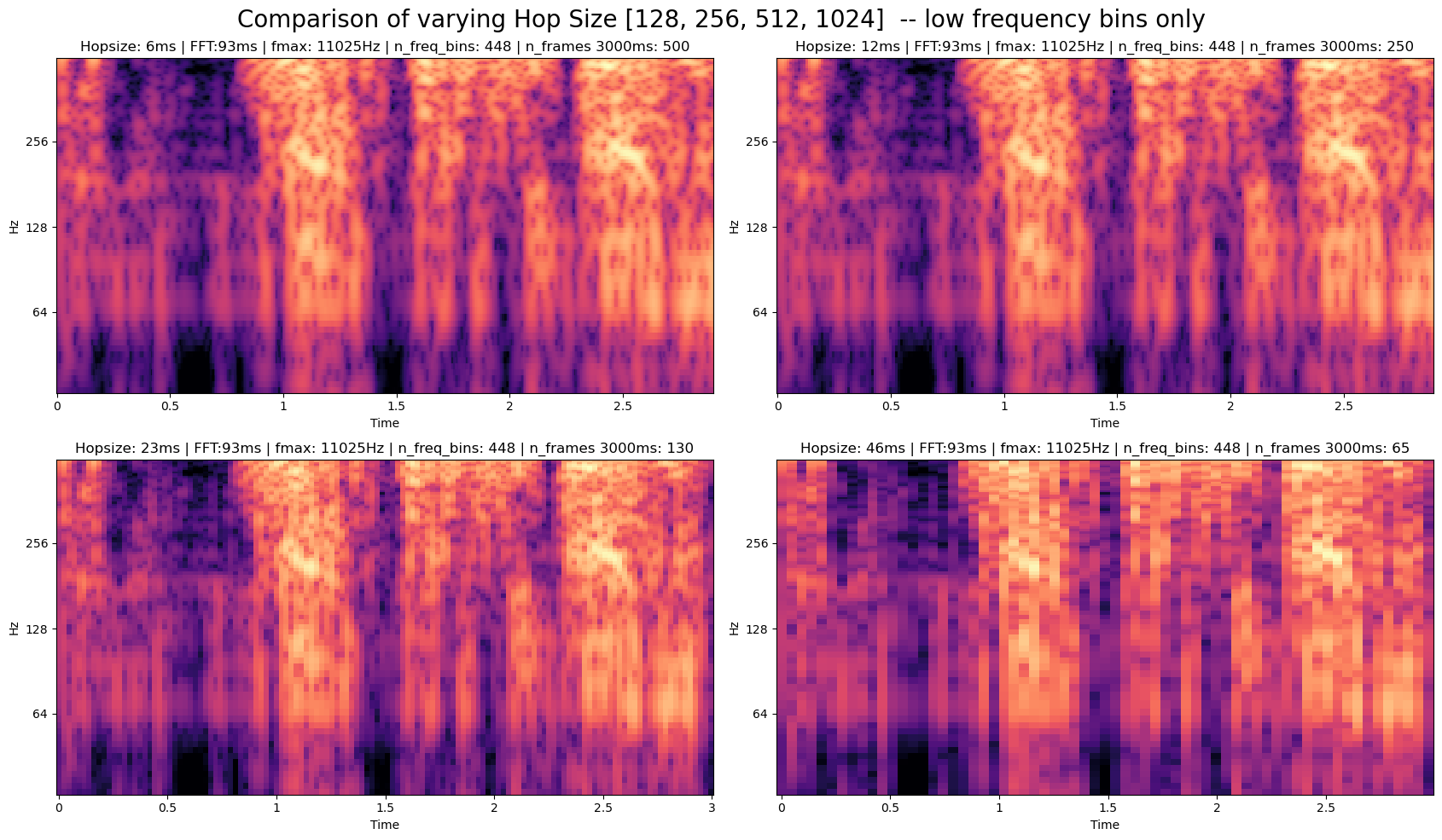
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Number of Time Frames | | |
| **200 Frames** | **400 Frames** | **600 Frames** |
|  | **6ms/128 samples** | 1.1s | 2.3s | 4.6s |
| Hop size | **12ms/256 samples** | 2.3s | 4.6s | 6.9s |
| **23ms/512 samples** | 4.6s | 9.2s | 13.8s |
| **46ms/1024 samples** | 9.2s | 18.4s | 27.6s |

Having a large hop size means we can train a model with a large portion of the total signal at once, but we cannot capture quick temporal changes very well. Also, it may have implications for the MIL algorithm. If a single input can capture the whole duration of the recording, there probably is no need for MIL. But again, the temporal features, which may be very important, may get lost). This could suggest, that without the MIL algorithm, the larger hop size may perform better because it captures more of the recording duration, but with MIL, the higher temporal resolution may give the better MIL results (yet possibly bad regular results).

A picture containing diagram, plot, line, screenshot

Description automatically generated





A picture containing screenshot, colorfulness, rectangle, pattern

Description automatically generated

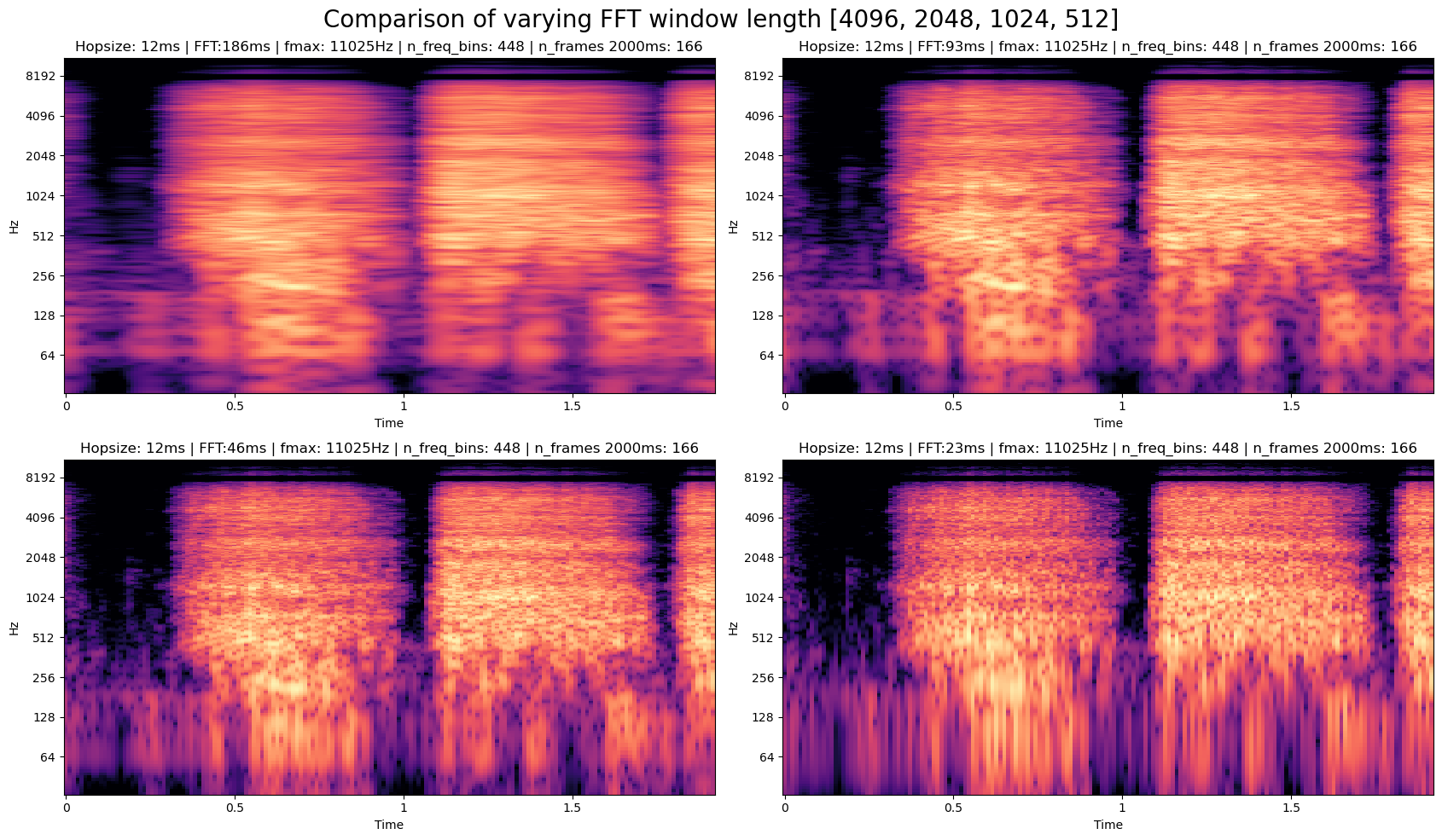
## Results:

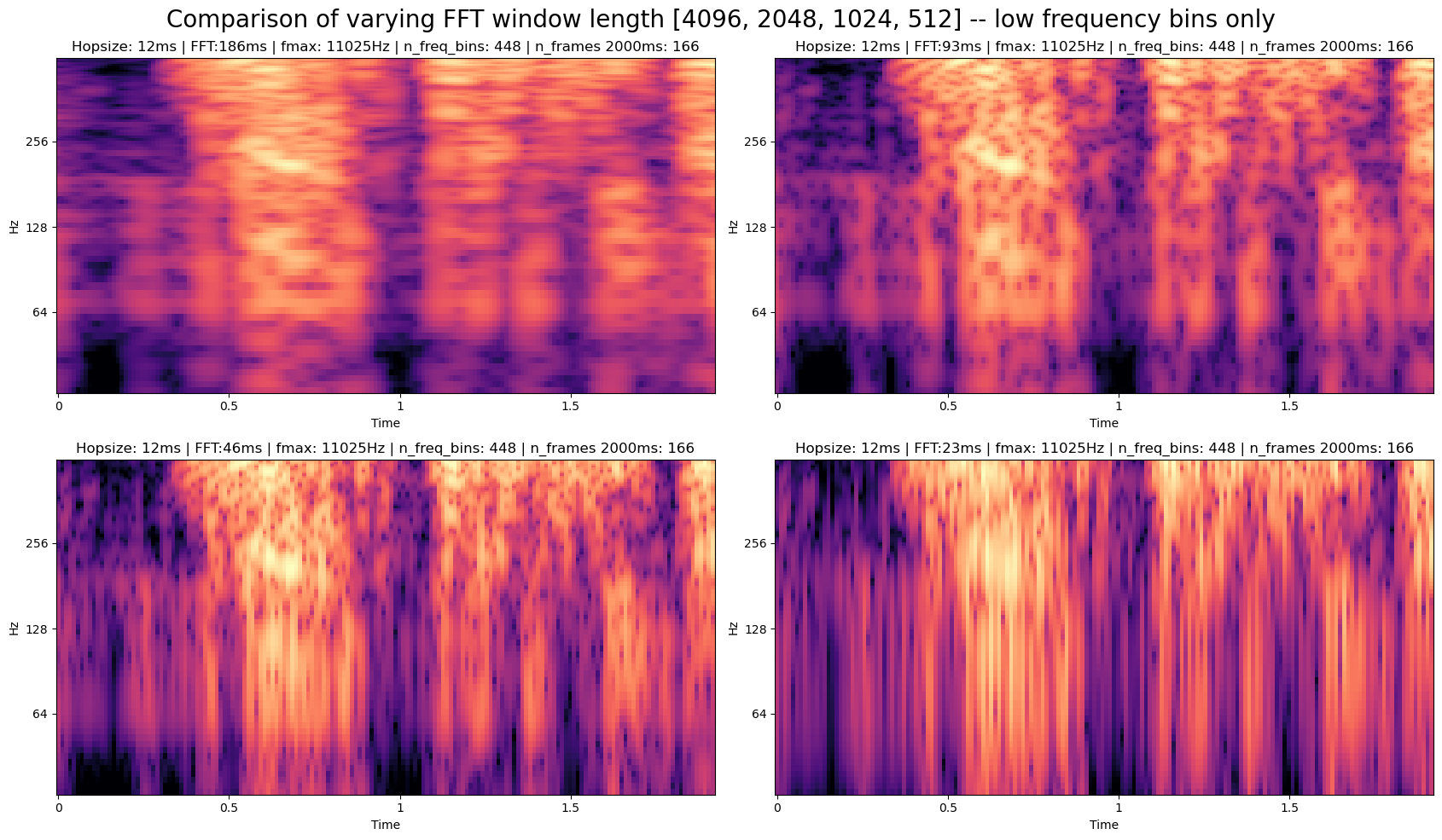
46ms hopsize seems excessively long and there does not seem to be any transitions but discrete jumps. 23ms also seems a bit jumpy but it is a lot better already. 12ms seems to be quite smooth already and going down to 6ms does not seems to improve that much. 12ms is chosen to move forward with.

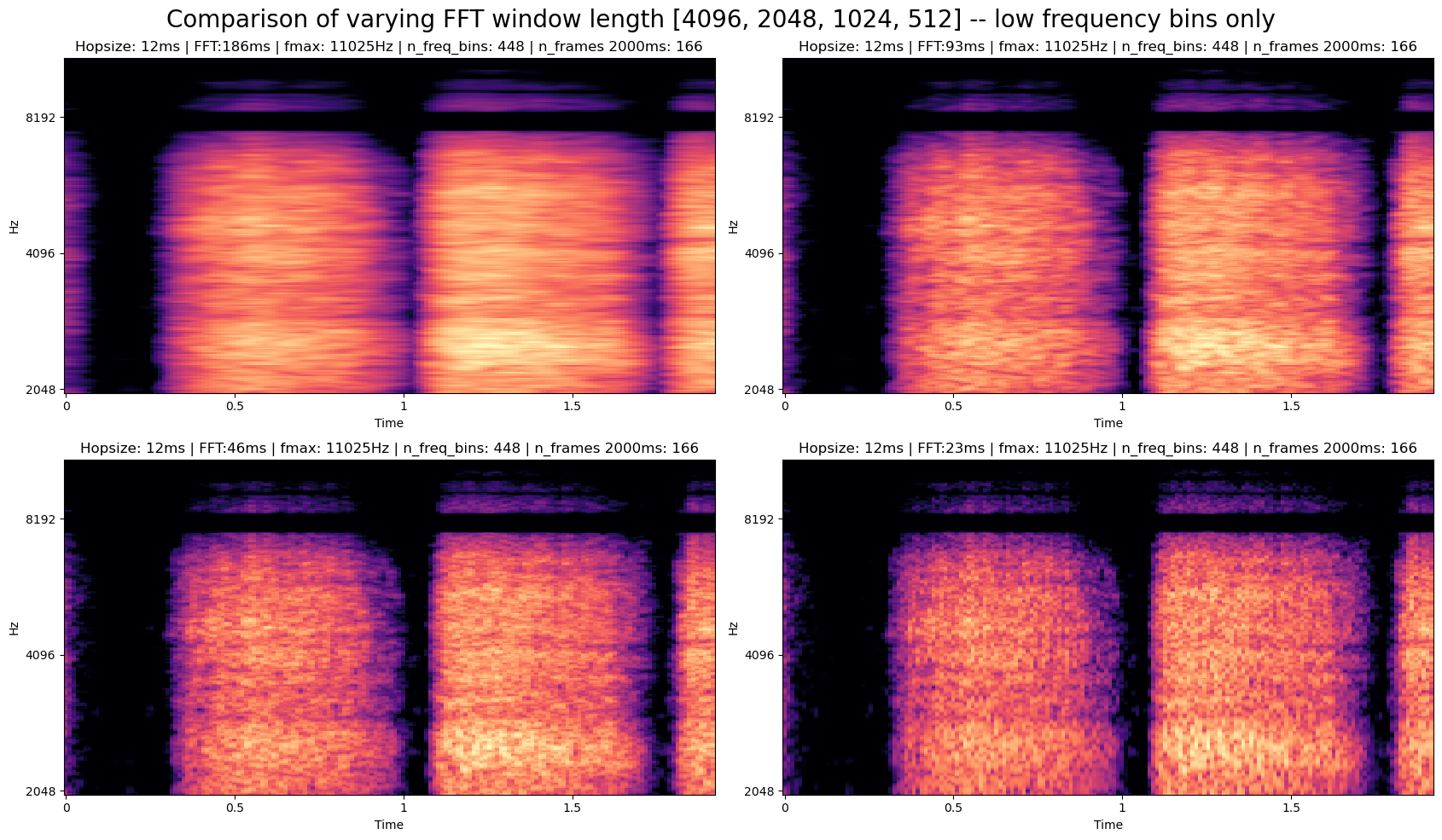
## FFT Length:

FFT length is responsible for averaging the frequency contents over time. A short FFT can capture temporal changes well but there may also be a lot of noise and variance with a short window. If the FFT length is equal to the hop size, there is no overlap which is probably not great as the overlap creates a smoother progression. The range of the FFT length we want to test will be between 512 and 2048 which corresponds to 23-92ms. This means that the window contains the wavelength of a ~50Hz (23ms window) to a 10Hz (92ms window) signal.

As a reasonable starting point, a 50% overlap will (probably) be tested. So if the FFT length is 92ms, the hop size is half of it, so 46ms.







## Results:

23ms is clearly too small a window in the low frequency range. It gets vertically completely blurred. On the other hand, the other extreme means 186ms and a horizontal blur in the high frequency region. The best choice probably lies between 1024 and 2048 samples. Although 512 seems to be a good choice looking only at the higher frequencies. There is an argument to be made to use varied lengths (like in the wavelet transform??)

# Favourite Choices:

As the number of frequency bins 224 was chosen, even though doubling them improved the result a bit, the trade-off of having twice as much data was not quite worth it(?). The choices for Hopsize were 128 and 256 samples while for the FFT window length they were 1024 and 2048. The most promising result appeared to be Hopsize=128 and FFT=1024. Which corresponds to an overlap of 87.5%. There is a lot of “redundant data” which is counterintuitive.

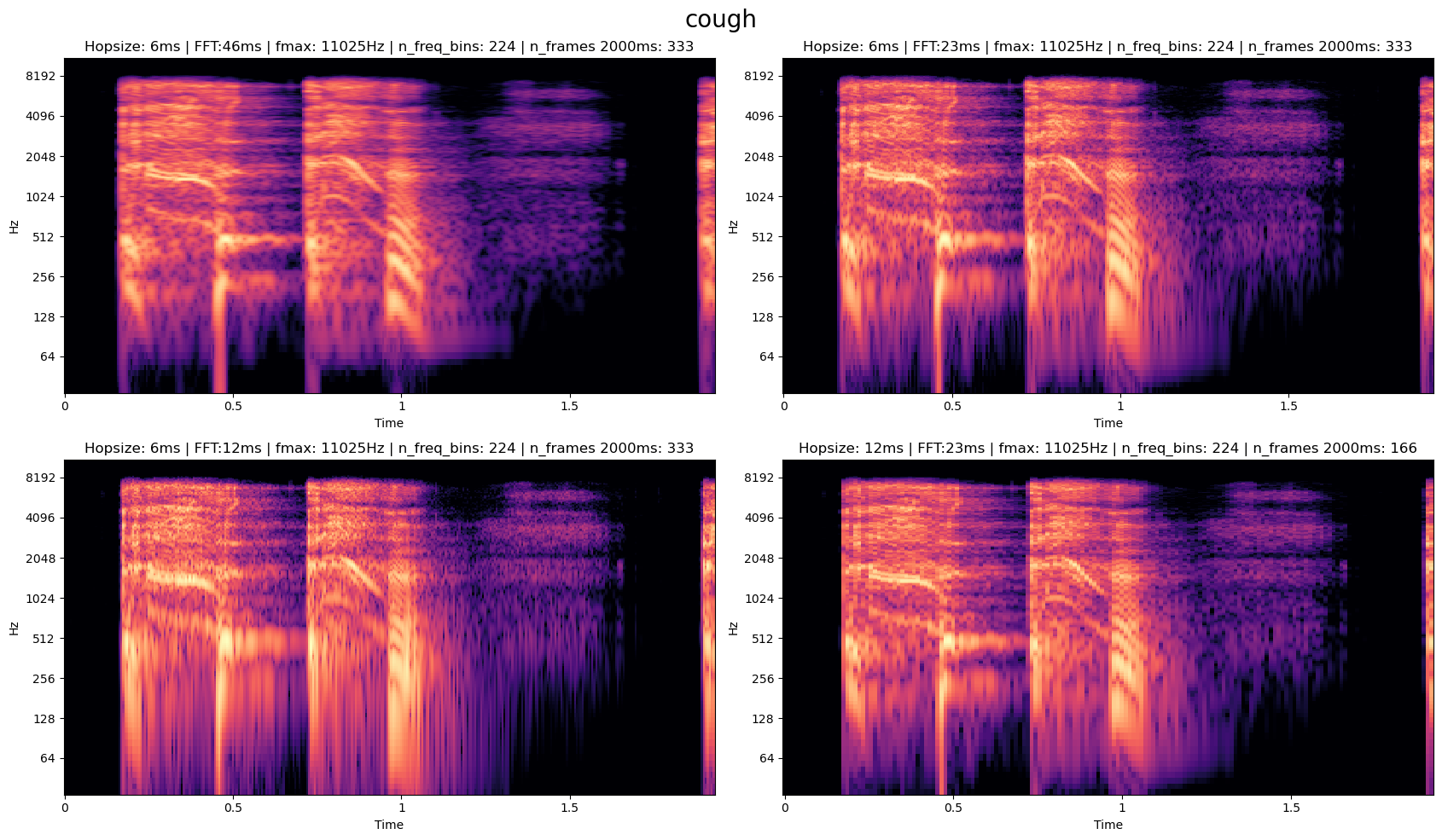
A picture containing screenshot, colorfulness

Description automatically generated

Min 6ms max 92ms hop size (92ms hop size ~= 20sekunden bei 224 frames)

# OTHER RECORDING TYPES:

# COUGH:



# Vowels

