Audio Speech Recogniser
Robin Rai 100242165

Contents

References

Loweimi, E., Ahadi, S. M., Drugman, T., and Loveymi, S. (2013). On the importance of pre-emphasis and window shape in phase-based speech recognition. In Drugman, T. and Dutoit, T., editors, *Advances in Nonlinear Speech Processing*, pages 160–167, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vaseghi, S. V. (2008). *Advanced Digital Signal, Processing and Noise Reduction*. John Wiley Sons.

Vergin, R. and O'Shaughnessy, D. (1995). Pre-emphasis and speech recognition. In *Proceedings 1995 Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 1062–1065 vol.2.

1 Speech Collection and Annotation

Our speech recognizer was to be trained to recognize 20 names. The training speech was to be recorded at as high of a quality as possible, while the test speech was deliberately varied. The training speech was recorded in batches of 20 names in random order with a purchased high-quality microphone. 40 of these recordings were made, recorded at 44.1KHz at 8 bits. For the testing data, 30 batches of 20 names were recorded in random order: 10 on two laptops with mild background noise each, and another 10 with the purchased microphone. Artificial noise could be introduced to these 30 batches to widen the variety of test scenarios further. During the live demonstration, the purchased microphone was used.

To train the Hidden Markov Model (HMM) used, label files must be produced. These tell the HMM what audio represents what words, for both training and evaluating. While the recording (and later the feature extraction) was automated via scripting, the labelling was done manually using SFSWin.

2 Feature extraction

Extracting Mel Frequency Cepstral Coefficients (MFCCs) was done through MATLAB. Several functions and scripts were used to automate our main feature extraction function. This function takes in a speech file, an optional noise file, and the number of dimensions to be created for each feature vector, and outputs an MFCC vector array and correctly sized Prototype file ready to be used with HMM.

The function first starts by checking and converting the inputted audio data to 16KHz, for consistency with all data including the noise data. The audio was then split into frames, each frame being 20 milliseconds long – an ideal value for quantizing speech. The frames were then overlapped by 50%. This was done so there were more areas of audio/speech change that could be modelled. Each frame was also hammed, to smooth the transition of amplitude from frame to frame.

2.1 Pre-emphasis

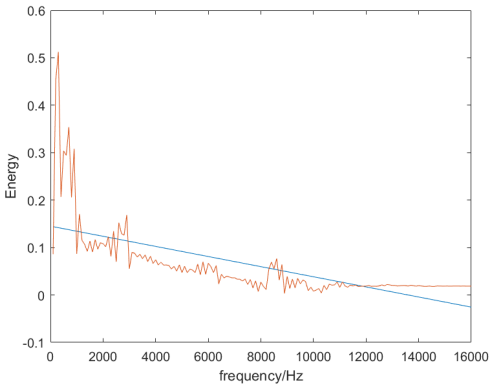

Figure 1: Before Pre-emphasis

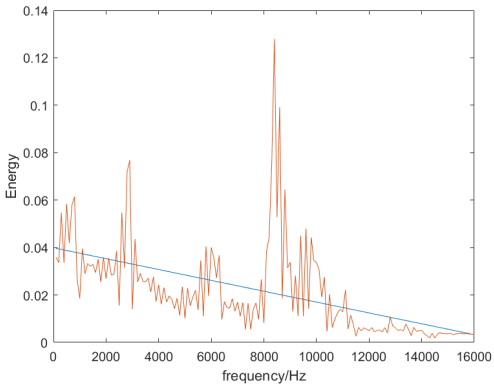

Figure 2: After Pre-emphasis

Figure 1 shows the magnitude spectrum of a speech frame sampled at 16 kHz. Observably, most of the energy is concentrated towards the lower frequency range. Linear approximation also confirms the energy decay. Pre-emphasis increases the energy content of higher-frequency regions of speech so that they

become stronger and more intelligible. Figure 2 shows this energy redistribution in action. However, the downside is that noise occurring at higher frequencies can also be boosted by an equal amount. To mitigate this effect, we perform spectral subtraction (see section x for discussion) prior to pre-emphasis remove as much noise as possible. The pre-emphasis filter can be applied to a signal s using: $s(t) - \alpha * s(t-1)$. We chose α (filter coefficient) to be between 0.94 and 0.97 as it is considered a sensible value in literature including (Loweimi et al., 2013) and (Vergin and O'Shaughnessy, 1995)

2.2 Discrete Fourier Transform (DFT) implementation

We use MATLAB's inbuilt FFT function to convert each frame from the time-domain to the spectral domain. However, we endeavoured to create our own DFT function using a technique called vectorization to speed up the execution time of our original DFT, previously implemented using a nested for-loop. In summary, we construct a Fourier transform matrix and multiplied with our frame to get the result of the DFT operation. We did not use our own DFT as it performed much slower than FFT.

2.3 Filter bank

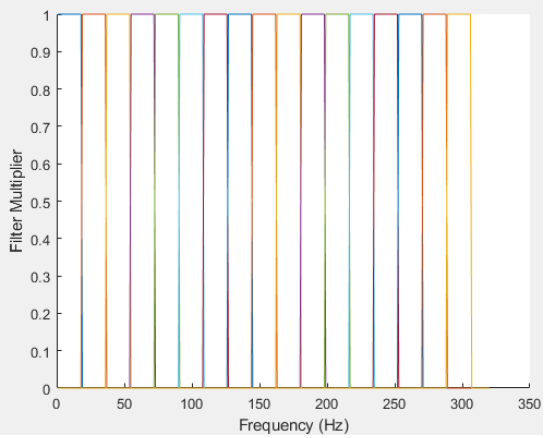

Figure 3: Linear Spaced Rectangular

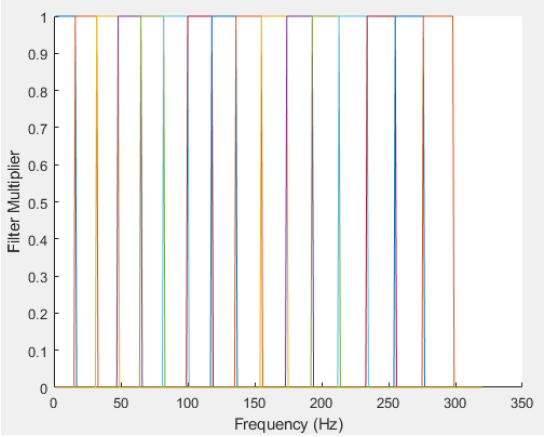

Figure 4: Mel Spaced Rectangular

A filter bank is a method of splitting each frame's data further into segments, so the HMM can train on differences and changes between them. Multiple methods were made and tested. Both triangular and rectangular shapes were used, along with linear and Mel spacing, Mel being more suited to the human ears' response to sound - lower frequencies are prioritized. The Mel spacing was formed by taking linearly spaced indexes in the Mel scale space, then converting them back into frequency. The rectangular filter shapes used a binary array, while the triangular system used a double array where each index of a Mel space was used as the left and right sides of an equal triangle, with the height being 1. Every value in between could then be calculated. Triangular shaped waves are used to emphasize and differentiate each bank from the other. Overlapping the triangles reduces data loss in between each triangle's peak, where the filter approaches 0.

We log the outputs of our filter bank function I.e., Mel coefficients to compress their amplitude effectively reducing the dynamic range. This compression makes features match more closely what human ears can hear as we do not perceive loudness linearly. In the log scale, the log of our speech signal can

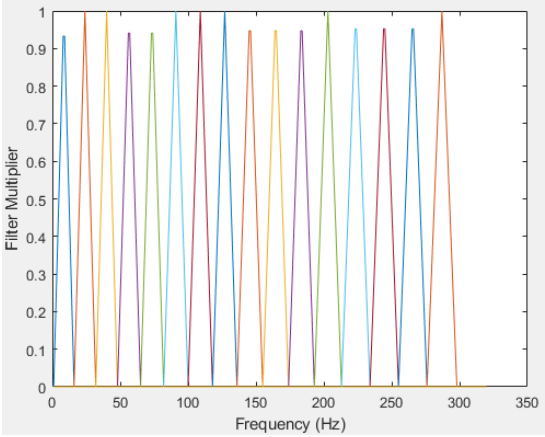

Figure 5: Mel Spaced Triangular

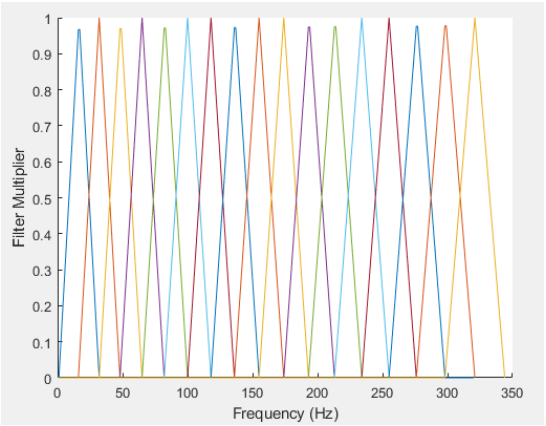

Figure 6: Mel Spaced Triangular Overlapped

be viewed as an addition of the gain, log of the excitation signal (pitch information) and log of the vocal tract filter.

2.4 Quefrency

By applying Discrete Cosine Function to the log of our Mel coefficients, we turn them from the spectral domain to the cepstral domain as shown. Pitch information is represented by high quefrency coefficients and is not important in English speech recognition. Hence, we discard the former by simply truncating the quefrency estimate by half retaining only vocal tract information.

2.5 Temporal Derivatives and Energy Component

Currently the HMM simply looks at features statically and separately. Adding temporal derivatives adds rate of change and rate of acceleration to the model - patterns of features changing over time. The velocity vector for a given feature vector can be calculated by simply measuring the difference between the next and previous vectors. This is repeated for the acceleration vector, but with the difference in velocity vectors rather than feature vectors.

Another extra data point to feed into the model is energy components. Each feature vector, or frame of speech, can have its energy measured and logged to reduce its influence on the model.

These extra data points can be appended onto each feature vector and result in the complete MFCC array ready to be written into a .mfc file ready for HVite. This increases the size of each vector by a factor of three, plus the energy component.

3 HMM Setup

Prototype files are used in combination with label files to initialise a probability matrix of state transitions for the HMM model to use. This prototype file is based on the number of transition states used. These number of states (and MFCC vectors) were varied and tested, with 18 states providing the best result. These prototype files were automated to be created, with a probability of 0.6 for staying on the current

state, and 0.4 for advancing states, bar the first and last states. These were chosen as they are reasonable probabilities.

The grammar file was setup in a way so that the HMM expected silence, then a name, then silence. The name and second silence were grouped and could be looped, for multiple words.

4 Noise Suppression

For noise suppression, we implemented spectral subtraction which was used in our best system for the live demonstration. We also experimented with Optimal Ratio masking to mitigate effects of artificially added noise.

4.1 Spectral Subtraction

Spectral subtraction is a noise suppression technique where we subtract an estimate of the spectral amplitude of noise from the noisy signal to obtain an estimate of the spectral amplitude of the signal.

According to (Vaseghi, 2008), we can express spectral subtraction as: $\hat{X}^b = \hat{Y}^b - \alpha\bar{N}^b$ where $\hat{X}^b$ is an estimate of the signal magnitude spectrum and $\bar{N}^b$ is the average magnitude of noise spectra. It is assumed that the noise is stationary i.e., energy is constant at any given point. Parameter α governs the amount of noise to be subtracted. In our case $\alpha = 1$ to have the best noise suppression possible.

The mean noise spectrum is obtained when there is no signal at which point only pure noise is present and is given as: $\bar{N}^b = \frac{1}{M}\sum_{i=0}^{M-1} N_i{}^b$

We assumed that first M=5 frames (100ms) represent pure noise. Implementation was as simple as looping over the first 5 frames, summing each frame's magnitude spectrum before computing the mean. We split the noisy signal x(n) into frames. Each frame is then overlapped by 50%. Before performing Discrete Fourier Transfer (DFT), each frame is Hanned to prevent spectral leakage – a phenomenon that can result in the loss of signals of smaller amplitude. Performing DFT on each frame gets us the magnitude and phase components. We then subtract the mean noise value from the magnitude component of each noisy frame to get back the clean magnitude. Finally, we go back to the complex spectrum by recombining the clean magnitude component and phase information and apply the Inverse Discrete Fourier Transform (IDFT) to return to the time domain. The full audio signal is then reconstructed by overlapping and adding all the frames back to a vector.

4.2 Optimal Ratio Masking

The second method of noise suppression is a modified version of Ideal Binary Masking (IBM). This involves using known and separate noise data that's been incorporated with speech. For each frame of audio a comparison between the noise and the speech data is made. If the speech data is louder, the resulting frame remains unmasked. If the noise is louder, the resulting frame is masked and silenced. Ratio masking is the same premise, but rather than having a simple binary method of masking, the resulting frame's volume is granularly quietened, based on the speech to noise ratio, allowing for smoother transitions between frames. Each frame was also Hammed for smoother transitions.

5 Evaluation

Over the span of more than 60 hours of work, our speech recogniser reached a peak accuracy of 95%, using varied test data from three separate microphones. When using just the high quality microphone for testing, 100% accuracy was achieved in almost every test.

5.1 MFCC Vector Size

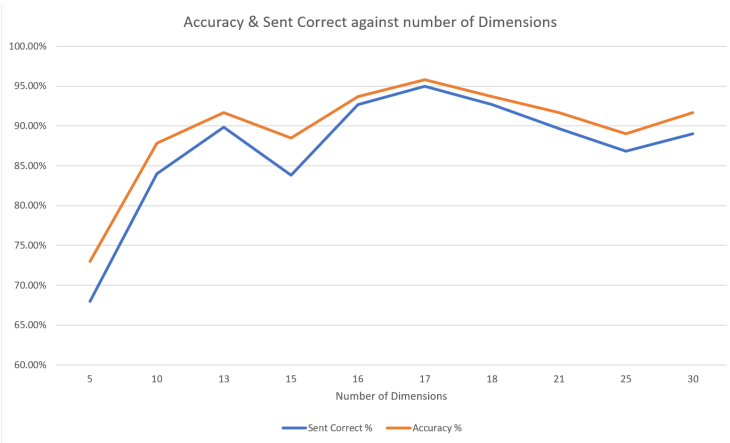

Figure 7: MFCC Vector size affecting quality

Our first test began with MFCC vector size. It is to be noted that the true dimension count is three times the number listed, plus one. This is due to the velocity and acceleration vectors. The highest accuracy and sent correct percent was with feature vector size of 17, or 52 dimensions. This was tested with multiple state counts and further tested and refined with 18 states, which was the highest accuracy achieving state count.

5.2 Number of States

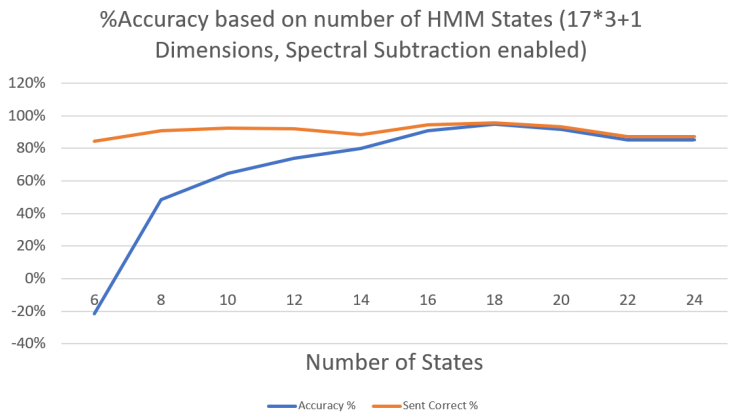

Figure 8: Number of states affecting quality

For number of states, accuracy increased up to a peak at 18, before beginning to fall again. The low accuracy from the lower number of states can be explained by each state covering too much area, while the decrease with larger numbers could be from each state not having enough data to work with effectively. It is to be noted that when using only high quality test data, 100% accuracy was obtained from state counts all the way from 5 to 24 states.

5.3 Filterbank Type

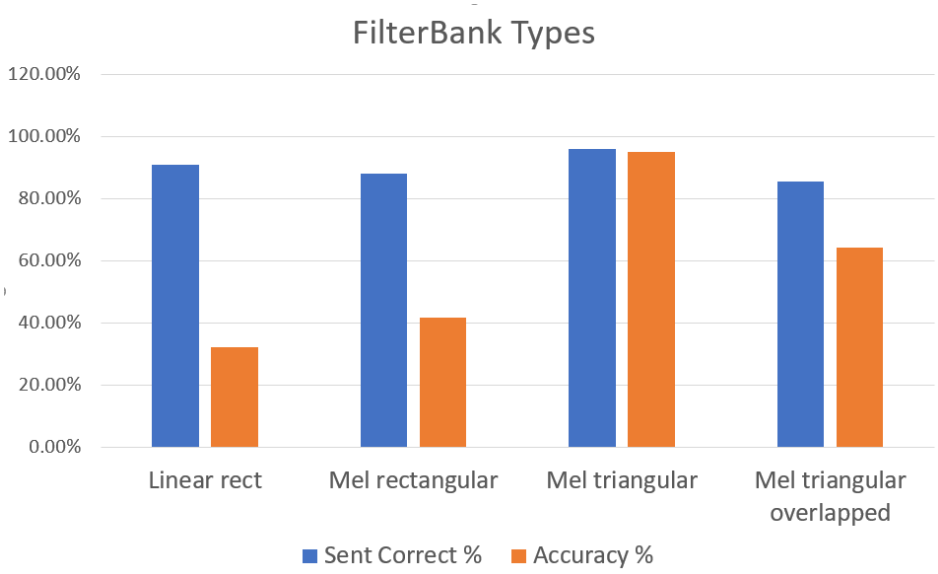

Figure 9: Filter bank type comparison

Four types of filter were tested, with linearly spaced rectangular waves, Mel spaced rectangular waves, Mel spaced triangular waves, and finally Mel spaced and overlapped triangular waves. The slight increase in accuracy between Mel and Linear spacing with rectangular waves is to be expected, as Mel spacing prioritises the lower frequencies of the speech, much like human ears. Looking back at Figure 4 the Mel spacing isn't very extreme at these lower frequencies also compared to the Linear spacing, correlating with the mild improvement. However overlapped triangular waves performing worse than triangles that aren't overlapped is perplexing. The triangular waves separate each feature from each other more, so it could be that the overlapping could reverse this, bringing the result closer to rectangular waves.

5.4 Temporal derivatives and the Energy Component

To test how temporal derivatives and the energy component affects accuracy, the highest result was taken utilising both the temporal derivatives and energy component (18 states with a feature vector size of 17/52 total dimensions). The conditions were then replicated and rerun, firstly without just the energy component, then without just the temporal component. The energy component provided an expected small increase in accuracy, while the temporal components increased accuracy by a fairly large margin. This could be due to the temporal derivatives having so many data points per frame for the HMM to work with, compared with the single data point per frame of the energy component.

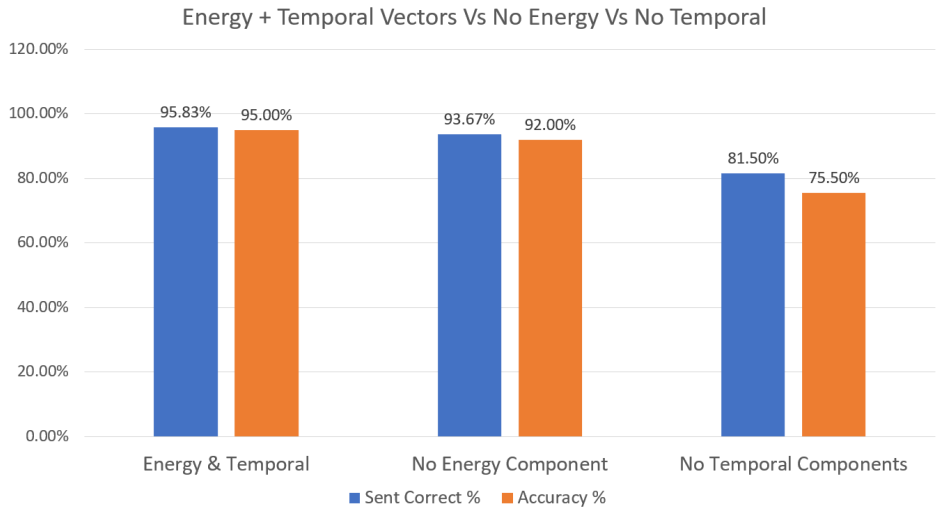

Figure 10: Temporal derivatives and energy component

5.5 Spectral Subtraction Testing

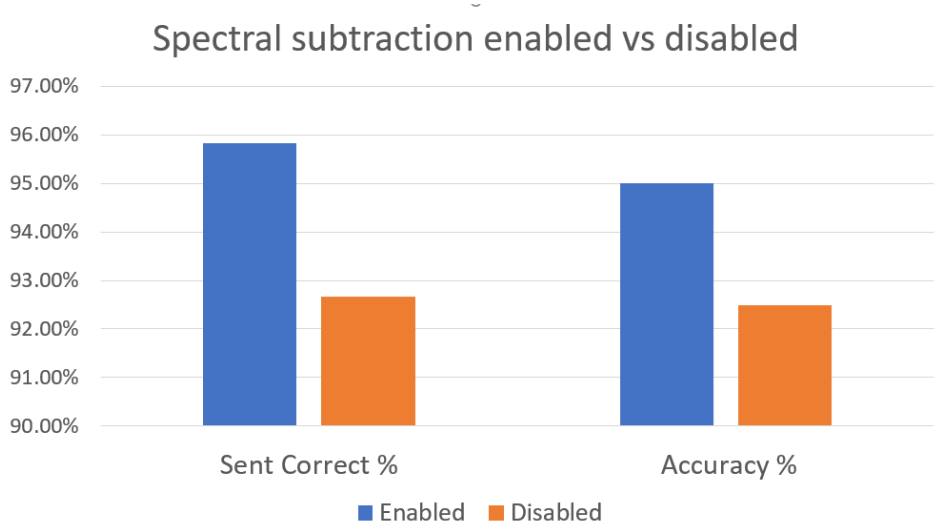

Figure 11: Spectral subtractions

Using spectral subtraction saw a moderate but welcome improvement with accuracy and sent correct %. This is likely from it working well to remove the mild and constant noise from the test speech recorded on the laptop microphones.

5.6 Optimal Ratio Masking testing

For optimal ratio masking, three types of artificial noise were added to speech and tested with and without the masking enabled. Babble noise from background conversations, factory noise, and distant machine gun noise was used. For both babble and factory noise, both accuracy and correct %s were very low. This

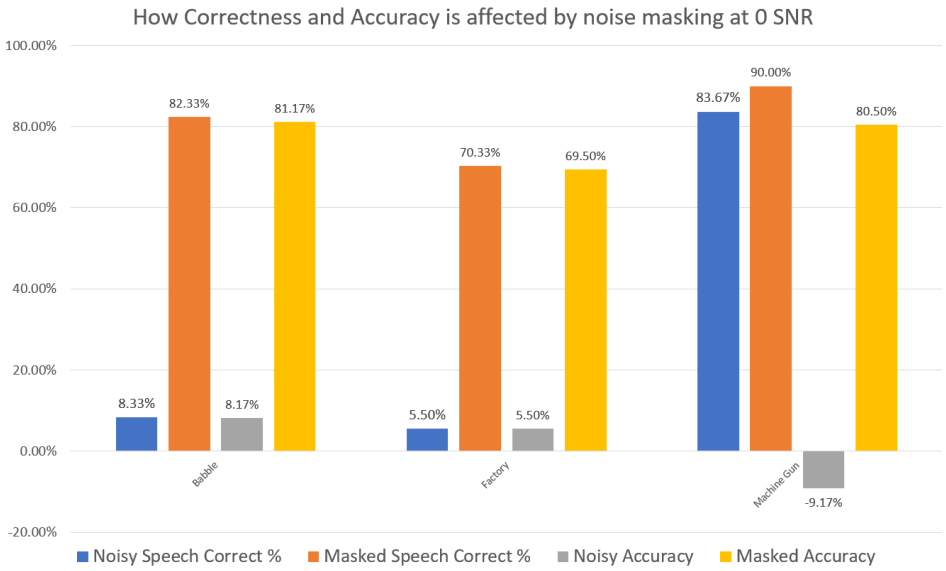

Figure 12: Noisy and filtered results with different types of noise

is likely from the noise being of a constant and high amplitude with varying content. The machine gun noise however kept correctness percents very high (Control results without noise were 95.83 correct % and 95% accuracy), while doing horribly with accuracy. These results are expected. Given the noise is very infrequent, there were relatively few cases where it actually added any noise to the names/speech, and when it did it was for a very brief moment, explaining the high correct %. The very low accuracy can be explained by the recogniser recognising the gunfire as names during silence and therefore creating a very high insertion count. When masking was enabled results were excellent, drastically raising accuracy and sent correct %s as well as coherency during playback. This was tested at multiple signal to noise ratios, including -10dB, where the unmasked audio is truly incoherent and lost to noise.

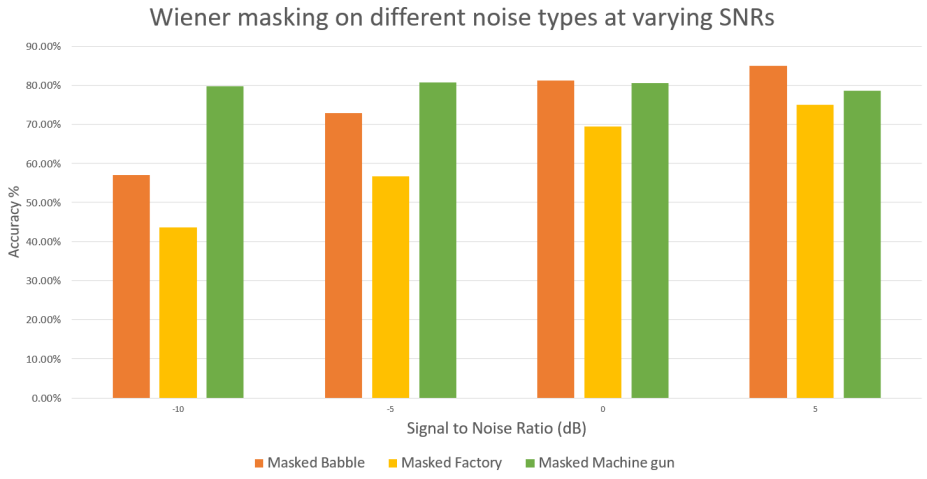

Figure 13: Wiener Ratio masking on different noise types and SNRs