Self Supervised Learning for Noise-Robust Key Word Spotting.

AALBORG UNIVERSITY
STUDENT REPORT

P9 Project
Mathematical Engineering

Authors:
Jacob Mørk

Supervisor:
Zheng-Hua Tan
Christophe Biscio

November 3, 2023

Department of Mathematical Sciences
Skjernvej 4A
Aalborg University
http://www.math.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Self Supervised Learning for Noise-Robust Key Word Spotting.

Theme:

Project Period:
fall-semester 2023

Project Group:
jmark18@student.aau.dk

Authors:
Jacob Mørk

Supervisors:
Zheng-Hua Tan
Christophe Biscio

Number of Pages: 12

Date of Completion:
November 3, 2023

Abstract:

□

jmark18@student.aau.dk

Jacob Mørk

Preface

This project has been written in the period from September 1st, 2023 to December 25th, 2023 by one Mathematical Engineering students at Aalborg University the third semester of the master's. I would like to thank supervisors Zheng-Hua Tan and Christophe Biscio for their support throughout the project period. I would also like to thank Holger Severin Bovbjerg for the additional help throughout the duration of the project.
In this project, it is assumed that the reader has an understanding of the subjects presented during study for the bachelor's and master's degrees in Mathematical Engineering from Aalborg University.

Aalborg University, November 3, 2023

Contents

1 | Introduction

Today keyword spotting (KWS) is implemented in various systems like voice assistants. For these systems to achieve satisfying performance the models typically rely on a large amount of labeled data. This is limiting these application to situations where only this kind of data is available. To accommodate for this problem [Holgers article] has been looking into using self supervised learning (SSL).

Self supervised learning (SSL) have existed for a while, and while it has been the same idea driving the SSL in different fields, the implementations has differed [source: Data2vec article, abstract]. Therefor [Data2vec article] has introduced, what they call data2vec, which is a *framework that uses the same learning method for either speech, NLP, or computer vision.*

Since most systems using KWS are implemented places with some or more background noise. If you use speech commands on your phone the background noise could be anything from a bus stopping to people speaking.

In today's society, it is almost a given that new technology have some level of voice assistants, whether it is simple voice commands or integrated voice assistants, like Apple's Siri or Google's Assistant, keyword spotting is a part of it. For the voice assistants, there is an activating keyword. Using a keyword, it is possible to avoid running a more computational expensive automatic speech recognition (ASR) when it is not needed. Keyword spotting (KWS) also has many other applications, such as speech data mining, audio indexing, and phone call routing [KWS overview].

2 | Keyword Spotting

Keyword spotting (KWS) has developed a lot over the years, with one of the earliest approaches being based on the use of large-vocabulary continuos speech recognition (LVCSR) systems. An advantage to LVCSR-based KWS is its flexibility to deal with changing/non-predefined keywords. However a big weakness to the LVCSR-based KWS is that it requires a large amount of computational resources, which can result in latency and this is especially an import factor when working with smaller online devises such as smartphones and smartwatches. A lighter alternative to LVCSR-based KWS is the keyword/filler hidden Markov model (HMM) approach. This approach trains a keyword HMM and a filler HMM to model keywords and non-keywords audio segments, respectively. To find the optimal path in the decoding graph viterbi decoding is applied at runtime. Viterbi decoding can end up being computational demanding, depending on the topology of the HMM. The KWS system is then triggered whenever the likelihood ratio of the keyword model versus filler model is larger than a set threshold [López-Espejo et al.: 2021].

In newer time deep neural networks (DNN) have become very popular and are being used to develop new KWS systems. These systems are referred to as deep KWS systems and have become the new state of the art in KWS. Deep KWS systems does not need any complicated search algorithms like viterbi decoding, and it comes with a huge improvement over the keyword/filler HMM approach in the case of low computational complexity [López-Espejo et al.: 2021]. In general deep KWS systems consist of tree steps, speech features extraction, DNN acoustic model, and posterior handling, which is illustrated in Figure 2.1.

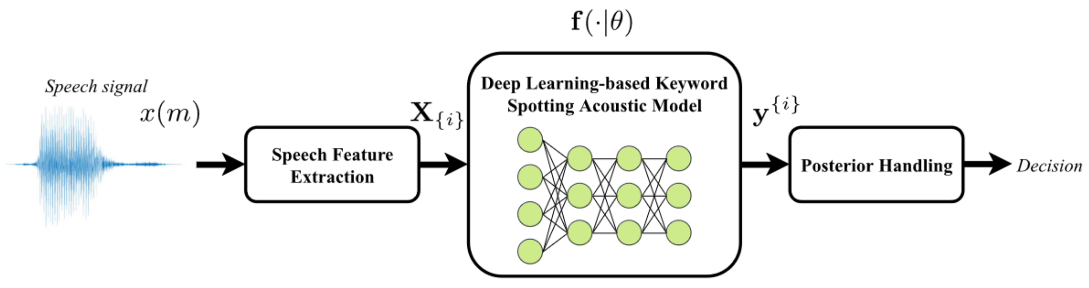
$\mathbf{f}(\cdot|\theta)$
Speech signal
$x(m)$
$\mathbf{X}_{\{i\}}$
Speech Feature Extraction
Deep Learning-based Keyword Spotting Acoustic Model
$\mathbf{y}^{\{i\}}$
Posterior Handling
Decision

Figure 2.1: Deep KWS system, from [López-Espejo et al.: 2021]

2.1 Speech Feature Extraction

The speech feature extractor converts a speech signal, $x(m)$, to a more compact representation of the speech. The speech features are often represented as a two dimensional matrix,

$$\mathbf{X} = (\mathbf{x}_0, \ldots, \mathbf{x}_{T-1}) \in \mathbb{R}^{K \times T}, \tag{2.1}$$

where T is the amount of feature vectors, and K is the size of each feature vector. The amount og feature vectors depends of the length of the speech signal [López-Espejo et al.: 2021].

Some of the popular speech features are the Mel-scale-related features. This could be the log-Mel spectral coefficients and Mel-frequency cepstral coefficients (MFCCs) [López-Espejo et al.: 2021]. The common way to extract the log-Mel spectral and the MFCC features is shown on Figure 2.2. On the figure it can be seen that to obtain the MFCC features you first have to get the log-Mel spectrogram coefficients, and then apply the discrete cosine transform to it. To stabilize, and speed up the training of the DNN acoustic model, the features are normalized to a mean of zero and a standard deviation of one. This will also make the model more generalized [López-Espejo et al.: 2021].

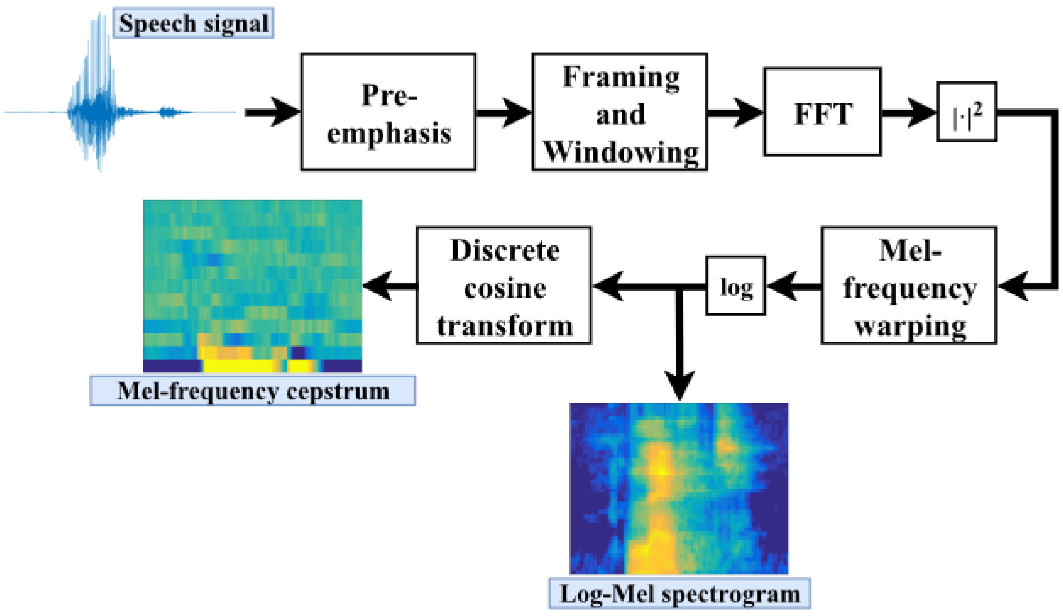

Figure 2.2: Classical pipeline for extracting log-Mel spectral and Mel-frequency cepstral speech features using the fast Fourier transform (FFT) [López-Espejo et al.: 2021]

2.2 DNN KWS acoustic model

The output of the speech feature extraction $\mathbf{X}$ is fed to the DNN acoustic model, which then outputs a sequence of probabilities, one for each keyword and non-keyword class. The way the acoustic model go through $\mathbf{X}$ is by taking a smaller section at a time, each section given as

$$\mathbf{X}_{\{i\}} = (\mathbf{x}_{is-P}, \ldots, \mathbf{x}_{is}, \ldots, \mathbf{x}_{is+F}) \in \mathbb{R}^{K \times (P+F+1)}, \tag{2.2}$$

where $i = \lceil \frac{P}{s} \rceil, \ldots, \lfloor \frac{T-1-F}{s} \rfloor$, is a integer segment, s represents the time shift i.e., the overlap, P is the number of past frames, and F the number of future frames. Two examples of how two consecutive sections can look like, can be seen on Figure 2.3.

Denoting the DNN acoustic model as

$$\mathbf{f}(\cdot|\theta) : \mathbb{R}^{K\times(P+F+1)} \to I^N, \tag{2.3}$$

where θ is the input of the model, $I = [0, 1]$, and N the number of classes, we get the output probabilities, $\mathbf{y}^{\{i\}}$, of the model, given a segment $\mathbf{X}_{\{i\}}$,

$$\mathbf{y}_n^{\{i\}} = \mathbf{f}_n\Big(\mathbf{X}_{\{i\}}|\theta\Big), \quad n = 1, \ldots, N, \tag{2.4}$$

where n denotes the n'th element of a vector. Denoting C_n as the n'th class we get that $\mathbf{y}_n^{\{i\}} = P(C_n|\mathbf{X}_{\{i\}}, \theta)$ is the probability of $\mathbf{X}_{\{i\}}$ belonging to C_n. To insure that $\mathbf{y}_n^{\{i\}}$ is a probability, i.e.,

$$\sum_{n=1}^{N} \mathbf{y}_n^{\{i\}} = 1 \quad \forall\ i, \tag{2.5}$$

a softmax activation function [Rothman: 2020] is commonly used on the last layer [López-Espejo et al.: 2021].

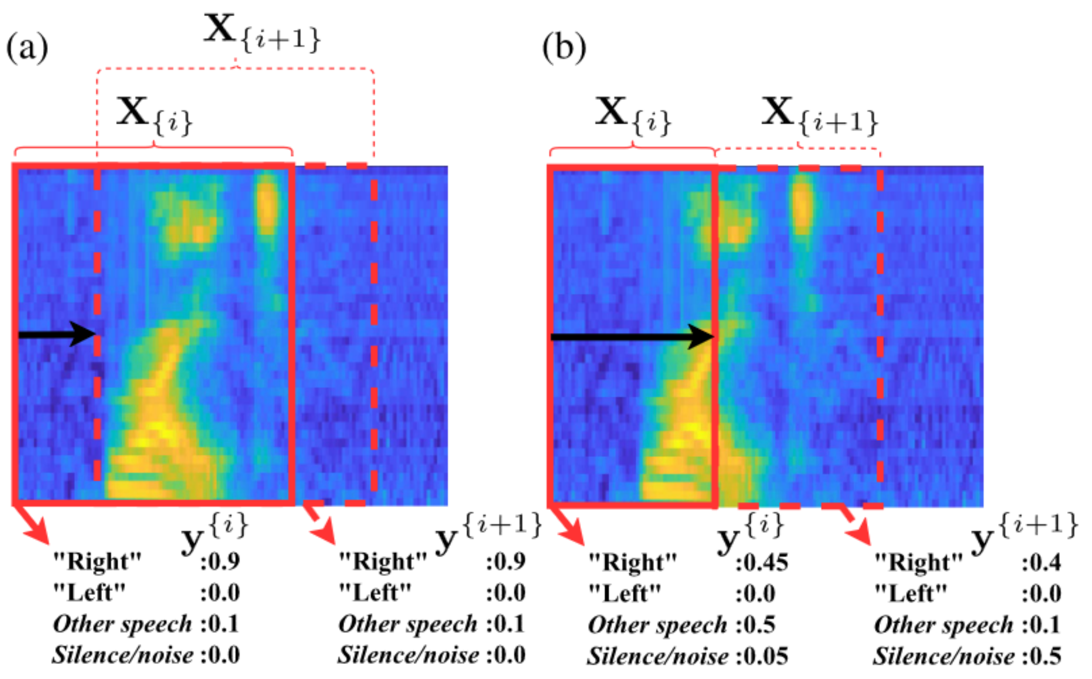

Figure 2.3: Example of the processing of two consecutive feature segments $\boldsymbol{X}_{\{i\}}$ and $\boldsymbol{X}_{\{i+1\}}$, from $\boldsymbol{X}$ comprising the keyword "right", by a DNN acoustic model: (a) when using an overlapping segmentation window, and (b) when using a smaller, non-overlapping one [López-Espejo et al.: 2021].

To pick which class a certain segment belongs to, one could just pick the class with highest probability, which could lead to different errors. Two cases where this approach will fail are shown in Figure 2.3, where an overlapping segment window is used on (a) and a smaller non-overlapping segment is used in (b). In the case of (a) the word "right" is found twice leading to a false positive detection and in the case of (b) "right" is not found an single time, leading to a false negative. This is one of the reasons why a good posterior handling is needed [López-Espejo et al.: 2021].

2.3 Posterior Handling

To get a good final decision of class, the sequence of probabilities, $\mathbf{y}^{\{i\}}$, needs to be processed. There are two main modes when applying posterior handling, non-streaming and streaming modes.

Non-streaming mode is the simpler of the two, it is a standard multi-class classification of independent input segments. Since it only works on one segment, it is needed for a segment to be long enough to contain a hole word, this could be around one second [López-Espejo et al.: 2021]. Posterior handling in this mode is then the straight forward methodclass of picking the with the highest probability. As mentioned above this is not a very realistic method, this is because KWS in not a static task [López-Espejo et al.: 2021].

Streaming mode is on the other hand when the processing happens continuos, and normally in real-time [López-Espejo et al.: 2021]. In this mode the output sequence of probabilities from the acoustic model

$$()\{\dots,\mathbf{y}^{\{i-1\}},\mathbf{y}^{\{i\}},\mathbf{y}^{\{i+1\}},\dots\}, \tag{2.6}$$

is typically smoothen over time, classically a moving average is used. The smoothed probabilities, $\bar{\mathbf{y}}^{\{i\}}$, are often used to determine the keyword, or the lack of one. This is done either by comparing them with a sensitive threshold or by picking the class with the highest probability within a time sliding window. In the case of two consecutive segments covering the same keyword, a simple method is implemented, where all spotted keywords are ignored for a short period of time right after a keyword is spotted.

3 | Self-Supervised Learning

3.1 Self-Supervised Learning

Self-Supervised Learning (SSL) is, unlike supervised learning, learning information about some data without any labels. The way SSL is not unsupervised is by defining a learning objective based on the underlying structure of the data itself, this is called a pretext task. In general this in done by predicting any unobserved or hidden part of the input data using only the observed or unhidden part of the input data. For natural langues processing (NLP) one way to do so is by hide words and try to predict the hidden words. a common way to do so is using masks. When using masks the objective is to predict the masked word from the surrounding words. This encourage the model to learn the relationships between different words, which then can be utilized in a range of downstream tasks. This could for example be translating text, summarizing, or generating text. In computer vision (CV) the same approach, predicting masked patches of an image or representation, have been used in models such as MAE and BYOL [KILDER]. Other approaches are also common in CV, for example using the objective of mapping two versions of the same image to the same representations. Even though the idea of SSL is very similar across different domains, such as NLC, CV, or audio, their algorithms and objectives are very different. This is mainly because they were developed with a single domain in mind. A framework which is created with multiple domains in mind is data2vec.

Maybe add some from data2vec article into
This is mainly from Meta Cookbook

3.2 Data2vec

Data2vec is a framework which is created to get closer to one leaning method for all SSL problems. The version of data2vec which is used in this project has implemented a method which works on either speech, NLP, or CV. The method used combines masked prediction with the learning of latent target representations generalized by using multiple network layers as targets. This is done by having a student mode and a teacher mode. First the input data is masked and encoded to create a representation of the masked input. The unmasked input data is then encoded and parameterized as an exponential moving average to create the training targets. Now the learning objective is for the student to predict the target representations, given a partial view of the input. This is illustrated on Figure 3.1.

The way the model is trained is to predict the representation of the unmasked sample, only knowing the encoding of the masked sample.

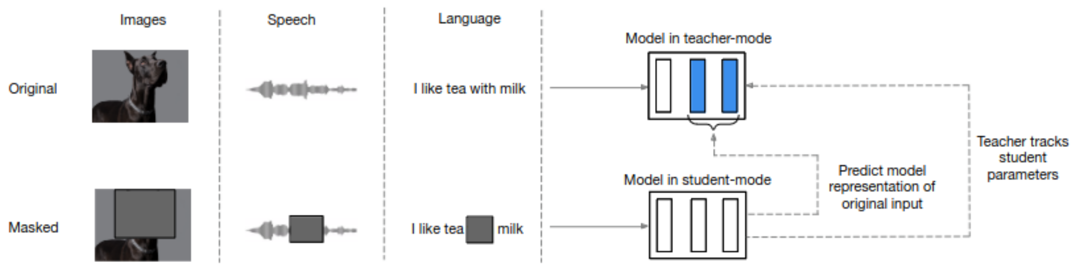

Figure 3.1: An illustration of how data2vec works on different data types, using the same process. The model creates a representation of the original input data and then a masked version of the input.

In the teacher mode the encoding is parameterized by an exponential moving average of the model parameters where the weights of the model in target-mode Δ are

$$\Delta \leftarrow \tau\Delta + (1 - \tau)\theta, \tag{3.1}$$

where θ is the model parameters, and τ is linearly increasing from τ_0 to τ_e over the first τ_n updates and are constant hereafter. This makes the teacher update more frequently in the beginning of the training when the model is more random and less frequently when good parameters have been learned.

The training targets are constructed from the last K blocks of parameters from the teacher network at the same time step as the student network. The training target at time step t, for a network with L blocks in total, is then

$$y_t = \sum_{l=L-K+1}^{L} \hat{a}_t^l, \tag{3.2}$$

where $\hat{a}_t^l$ is the normalized output of block l at time step t. The normalizing helps the network not to collapse and layers with high norm to not dominate the target feature. For speech the normalizing used is instance normalizing [Ulyanov et al.: 2016] without any learned parameters over the current input sample, this is because the neighboring representations are highly correlated due to small stride over the input data. Whereas for NLP and CV a parameter-less layer normalization [Ba et al.: 2016] is used.

4 | Results

In this chapter, the results of the test are presented. There are two levels to the test, one where the models are tested on keywords where the same kind of noise is applied, and one where more types of noise are added to the test set.

4.1 Only Testing on the Same Noise

Here the different models are tested using, a test set created of different sound files where the same kind of noise, as the model is trained on, is added. The results are presented in Table 4.1, where a subtable for each noise level is presented.

4.2 Testing on More Noise

		Data2vec pretraining	
snr-10	Baseline	non-noisy	noisy
KWT-1	0.3119	0.3445	0.3295
KWT-2	0.3007	0.3404	0.3487
KWT-3	0.294	0.3159	0.3595

(a) Snr -10

		Data2vec pretraining	
snr-5	Baseline	non-noisy	noisy
KWT-1	0.4672	0.5131	0.5028
KWT-2	0.4519	0.5139	0.5328
KWT-3	0.4348	0.4973	0.5706

(b) Snr -5

		Data2vec pretraining	
snr0	Baseline	non-noise	noisy
KWT-1	0.5936	0.6664	0.698
KWT-2	0.5899	0.6864	0.7178
KWT-3	0.5706	0.6181	0.7215

(c) Snr 0

		Data2vec pretraining	
snr5	Baseline	non-noise	noisy
KWT-1	0.6939	0.7534	0.7927
KWT-2	0.6824	0.7954	0.7969
KWT-3	0.6633	0.7498	0.7883

(d) Snr 5

		Data2vec pretraining	
snr10	Baseline	non-noisy	noisy
KWT-1	0.7383	0.8287	0.8368
KWT-2	0.7464	0.8415	0.8411
KWT-3	0.7302	0.8205	0.8537

(e) Snr 10

		Data2vec pretraining	
snr15	Baseline	non-noisy	noisy
KWT-1	0.7836	0.8418	0.8549
KWT-2	0.7675	0.8634	0.8725
KWT-3	0.7543	0.8345	0.842

(f) Snr 15

		Data2vec pretraining	
snr20	Baseline	non-noisy	noisy
KWT-1	0.8024	0.8610	0.8571
KWT-2	0.7965	0.8730	0.8737
KWT-3	0.7817	0.8379	0.8655

(g) Snr 20

Table 4.1

		Data2vec pretraining	
snr-10	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(a) Snr -10

		Data2vec pretraining	
snr-5	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(b) Snr -5

		Data2vec pretraining	
snr0	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(c) Snr 0

		Data2vec pretraining	
snr5	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(d) Snr 5

		Data2vec pretraining	
snr10	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(e) Snr 10

		Data2vec pretraining	
snr15	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(f) Snr 15

		Data2vec pretraining	
snr20	Baseline	non-noisy	noisy
KWT-1	–	–	–
KWT-2	–	–	–
KWT-3	–	–	–

(g) Snr 20

Table 4.2

5 | Discussion

6 | Conclusion

Bibliography

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450.*

Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., and Auli, M. (2022). Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, pages 1298–1312. PMLR.

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., et al. (2023). A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210.*

Bovbjerg, H. S. and Tan, Z.-H. (2023). Improving label-deficient keyword spotting through self-supervised pretraining. In *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pages 1–5. IEEE.

López-Espejo, I., Tan, Z.-H., Hansen, J. H., and Jensen, J. (2021). Deep spoken keyword spotting: An overview. *IEEE Access*, 10:4169–4199.

Rothman, D. (2020). *Artificial Intelligence By Example: Acquire advanced AI, machine learning, and deep learning design skills.* Packt Publishing Ltd.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022.*

A | Constraints