Self Supervised Learning for Noise-Robust Key Word Spotting.

AALBORG UNIVERSITY

STUDENT REPORT

P9 Project
Mathematical Engineering

Author:
Jacob Mørk

Supervisor:
Zheng-Hua Tan
Holder Severin Bovbjerg
Christophe Biscio

November 23, 2023

Written with pdfLaTeX.

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.

Abstract:

Theme:

Project Period:
fall-semester 2023

Project Group:
jmark18@student.aau.dk

Authors:
Jacob Mørk

Supervisors:
Zheng-Hua Tan
Holder Severin Bovbjerg
Christophe Biscio

Number of Pages: 18

Date of Completion:
November 23, 2023

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, Holger Severin Bovbjerg, and Christophe Biscio for their support throughout the project period.
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 23, 2023

Contents

1 Introduction **1**

2 Keyword Spotting **2**
2.1 Speech Feature Extraction 2
2.2 DNN KWS acoustic model 3
2.2.1 Transformer Models 4
2.3 Posterior Handling 5

3 Self-Supervised Learning **7**
3.1 Self-Supervised Learning 7
3.2 Data2vec 7

4 Experiment **9**
4.1 Data 9
4.2 Model Setup 9
4.2.1 Model Training 10
4.3 Test Setup 10
4.4 Results 11
4.4.1 Testing on BUS and BBL 11
4.4.2 Testing on BUS, BBL, CAF, and SNN 13

5 Discussion **17**

6 Conclusion **18**

Bibliography **19**

A Data **21**
A.1 Results 21

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. This is an import factor when working with smaller online devices 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. Viterbi decoding is applied at runtime to find the optimal path in the decoding graph. 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].

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 do not need any complicated search algorithms like viterbi decoding, and they come 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 three steps: speech features extraction, DNN acoustic model, and posterior handling, which is illustrated in Figure 2.1.

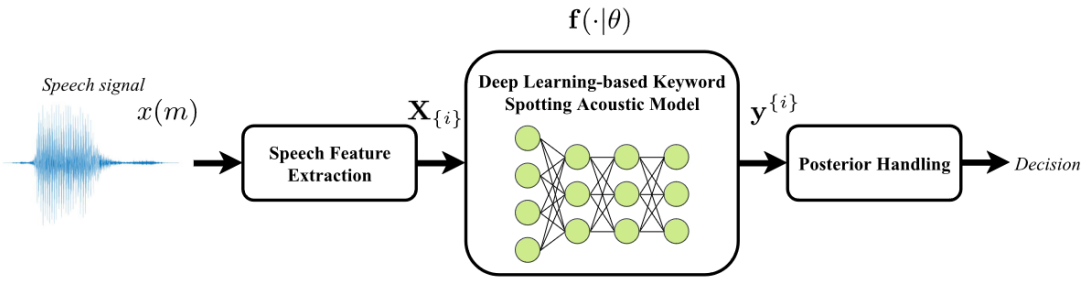

Figure 2.1: Deep KWS system, from [López-Espejo et al.: 2021]

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 of feature vectors depends on the length of the speech signal [López-Espejo et al.: 2021].

Some of the popular speech features are the Mel-scale-related features. These 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].

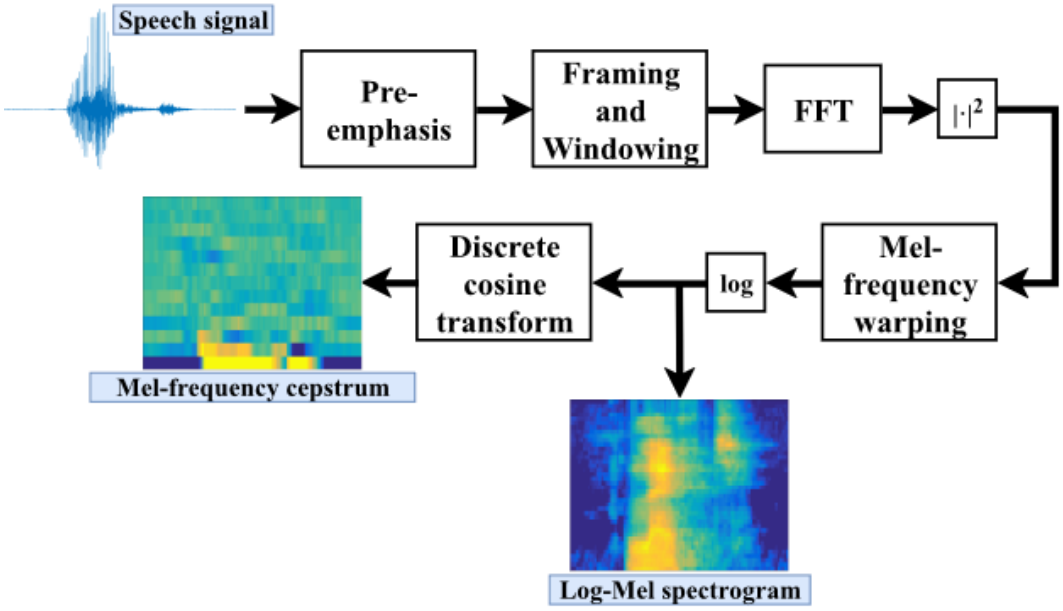

Figure 2.2: Classical pipeline for extracting log-Mel spectral and Mel-frequency cepstral speech features using the fast Fourier transform (FFT) [López-Espejo et al.: 2021]

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 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 consecutive sections are illustrated on Figure 2.3.

Denoting the DNN acoustic model as

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

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 ensure 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].

2.2.1 Transformer Models

Transformer models are one of the newer KWS models. The transformer model rely entirely on attention mechanisms using no convolutions or recurrence []. The transformer model apply self-attention mechanism to focus on different parts of the input. This is possible since it takes the full input sequence at ones, unlike RNNs. This also makes the transformer model more efficient to train since it allows for better parallelization [].

The idea behind the attention mechanism is to make the model able to learn which part og the input data to focus on. The self-attention mechanism applied in transformer models works by applying a Scaled Dot Product Attention to the input data which makes it possible, for the model, to attend to different parts of the input [].

A Scaled Dot Product get information from the source sequence with is relevant for the target sequence. Specifically, the target sequence would be encoded as $Q \in \mathbb{R}^{n\times m}$, and the source sequence as $K \in \mathbb{R}^{n\times m}$, $V \in \mathbb{R}^{n\times m}$, where n is the length of the target and source sequences, and m the is the dimension of the hidden dimension []. The Scaled Dot Product attention is then calculated as:

$$Att(Q, K, V) = \texttt{softmax}\left(\frac{QK^T}{\sqrt{m}}\right)V, \tag{2.6}$$

where $\texttt{softmax}(\cdot)$ is the row-wise softmax [].

Instead of just using self-attention, the transformer model uses multi-head attention, which is multiple sef-attention modules stacked, concatenated, and linearly projected to the expected transformer output dimension. The projections $W_i^Q \in \mathbb{R}^{m\times\frac{m}{h}}$, $W_i^K \in \mathbb{R}^{m\times\frac{m}{h}}$, $W_i^V \in \mathbb{R}^{m\times\frac{m}{h}}$, and $W^O \in \mathbb{R}^{m\times m}$

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2.7}$$

$$\text{where } head_i = Att(QW_i^Q, KW_i^K, VW_i^V) \quad i \in [1, \ldots, h] \tag{2.8}$$

The pros of using multi-head attention is that it allows the model to attend to different parts of the input, e.g., short-time dependencies and long-term dependencies [].

Combining the multi-head attention with a multilayer perceptron (MLP) results in a transformer encoder block, illustrated on REF.

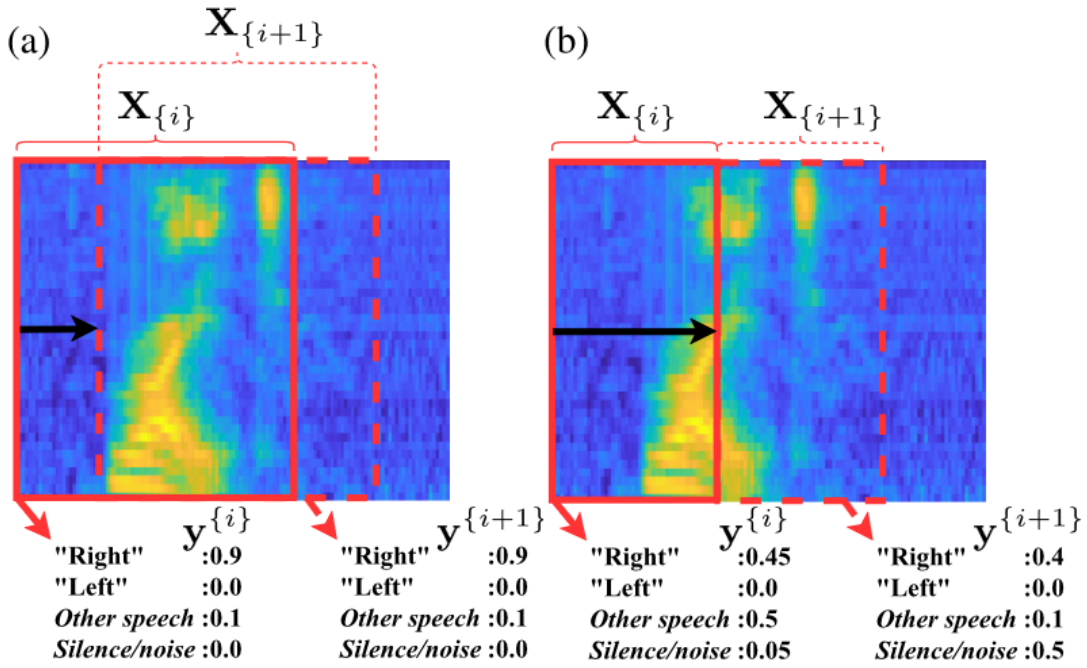

Figure 2.3: Example of the processing of two consecutive feature segments $\boldsymbol{X}_{\{i\}}$ and $\boldsymbol{X}_{\{i+1\}}$, from $\boldsymbol{X}$ comprising the keyword "right", by a DNN acoustic model: (a) when using an overlapping segmentation window, and (b) when using a smaller, non-overlapping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

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, the segment needs to be long enough to contain a whole word – this could be around one second [López-Espejo et al.: 2021]. Posterior handling, in this mode, is the straight forward method of picking the class with the highest probability. This is not a very realistic method, because KWS in not a static task [López-Espejo et al.: 2021]. 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 on (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 a 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].

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

$$()\{\dots, \mathbf{y}^{\{i-1\}}, \mathbf{y}^{\{i\}}, \mathbf{y}^{\{i+1\}}, \dots\}, \tag{2.9}$$

is typically smoothened 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.

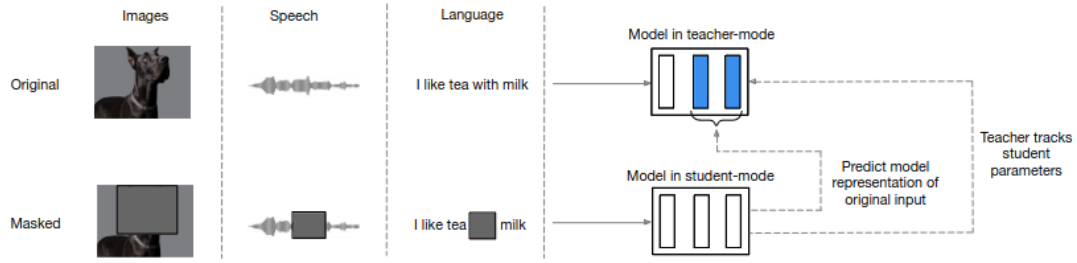

Figure 3.1: An illustration of how data2vec works on different data types, using the same process. The model creates a representation of the original input data and then a masked version of the input.

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 | Experiment

In this chapter the experiment will be outlined, whereafter the results of the experiment will be presented. The experiment includes five different models: two baseline models, and three test models.

4.1 Data

The data used in this project is the Google Speech Commands V2 dataset [Warden: 2018], which consists of 105829 one second recordings of 35 different keywords, outlined in Table A.1. The dataset has been manipulated such that 80% of the training data are without labels. The 80% will be used to pretrain the models, and the remaining 20% will then be used to finetune the models and to train a baseline model. The final spilt of the recordings can be seen in Table 4.1

Use	Pretraining	finetuning/ Baseline	Validating	Testing
Recordings	67874	16969	9981	11005

Table 4.1: How the 105820 recordings are split

Four different types of noise are used in the experiment, which can be put into two categories, one being stationary an another being non-stationary. The two types of noises in the stationary category are bus (BUS) and cafe (CAF), which have been used by [Barker et al.: 2015]. The other two types of noises are babble (BBL) and speech shaped noise (SSN), which were generated by [Kolboek et al.: 2016]. The noises have been added to the keywords in 7 different SNR's: -10, -5, 0, 5, 10, 15, 20. This yields 28 datasets, one for each noise with each SNR. For each SNR two mixed data sets have been created, one is a 50/50 mixture of keywords with BUS and BBL noise. Another mixed dataset is a equal mixture of keywords with the four types of noise.

4.2 Model Setup

The training and testing is using the same setup and models as in [Bovbjerg and Tan: 2023]. Here a keyword transformer (KWT) [Berg et al.: 2021], with a added mean of the encodings of each time step, is used.

Following [Bovbjerg and Tan: 2023], three versions of the KWT is used. Varying the transformers amount of attention heads from one to three and the encoder dimension from

64 to 192 yields the three models: KWT-1, KWT-2, and KWT-3, with $0.6 \cdot 10^6$, $2.4 \cdot 10^6$, and $5.4 \cdot 10^6$ parameters respectively [Bovbjerg and Tan: 2023].

In order to validate the noise-robustness of the SSL models two baseline models are trained. One being the baseline-supervised, which is a fully supervised model trained on the 20% labeled data. The second baseline model is a model pretrained and finetuned using noiseless data, this model is referred to as the baseline-Data2vec model. The three test models are all finetuned on the same data as the baseline-supervised is trained on and pretrained on different data. One is trained on noiseless data, and another is trained on noisy data, these are referred to as pretrained-noiseless and pretrained-noisy respectively. The last test model is trained by feeding the student noisy data and the teacher noiseless data, this model is referred to as the pretrained-both model.

Training the five models three times, one with each KWT results in 15 models in total.

4.2.1 Model Training

Using the same setup as in [Bovbjerg and Tan: 2023] the finetuning and baseline-supervised model is trained for 140 epochs with a batch size of 512, using cross entropy as the learning objective. The weights are updated using the AdamW [Loshchilov and Hutter: 2018] optimizer with a learning rate of $1 \cdot 10^{-3}$ and a weight decay of 0.1, where the learning rate schedular used is a 10 epochs linear warmup followed by cosine annealing. Furthermore, doing the training of the model SpecAugment [Park et al.: 2019] is randomly applied to mask blocks in both time and feature dimension [Bovbjerg and Tan: 2023].

For the pretraining the time domain masking strategy is the same as used in Wav2Vec []. Here a MFCC vector is picked with probability of 0.65 and the next 10 MFCC vectors are then replaced by a mask token embedding. The hyperparameters used are the same as the ones used in [Bovbjerg and Tan: 2023], where most of the hyperparameters have been chosen according to the original Data2vec study [Baevski et al.: 2022], with a few changes due to different in data and hardware.

The noisy data used to train the models is the data where only BUS and BBL noise is applied. This entails that non of the models have any experience with the noise types CAF or SNN.

The models have been trained using the implementation created by the author of [Bovbjerg and Tan: 2023].

4.3 Test Setup

To validate the models, the accuracy of the model , when testing them on new sound files which they have never seen before, is calculated. This is done using

$$acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.1}$$

on each keyword and average the results.

Two test are conducted, one where the the noise added to the sound files are the same kind of noise which is added to the training data (i.e., BUS and BBL). The results of this test can be found in Section 4.4.1. The second test is performed using the four noise types (i.e., BUS, BBL, CAF, and SSN). The results of this test can be found in Section 4.4.2.

4.4 Results

4.4.1 Testing on BUS and BBL

In this section the accuracy of the trained models when they are tested on new recordings with BUS and BBL noise added. On Table 4.2 the accuracies of the KWT-1 models are presented and on Figure 4.1 the accuracies are plotted for a more visual comparison.

	KWT-1				
SNR	Baseline - Supervised	Baseline - Deta2vec	Pretrained - Noiseless	Pretrained - Noisy	Pretrained - Both
-10	0.3459	0.3791	0.3784	0.3827	0.3996
-5	0.4805	0.5244	0.5324	0.5537	0.5744
0	0.6157	0.6641	0.6715	0.6909	0.7107
5	0.6925	0.7459	0.7558	0.7737	0.7856
10	0.7335	0.7801	0.7999	0.8086	0.8280
15	0.7617	0.8120	0.8172	0.8294	0.8509
20	0.7634	0.8155	0.8288	0.8383	0.8553

Table 4.2: Shows the accuracy of the five models trained using KWT-1, when tested on data with BUS and BBL noise added. The first two columns are the baseline test, a supervised and a SSL trained using only noiseless data. The last three are trained using different data in the pretraning, one uses noiseless, the second uses noisy, and the last uses noisy for the student and noiseless for the teacher.

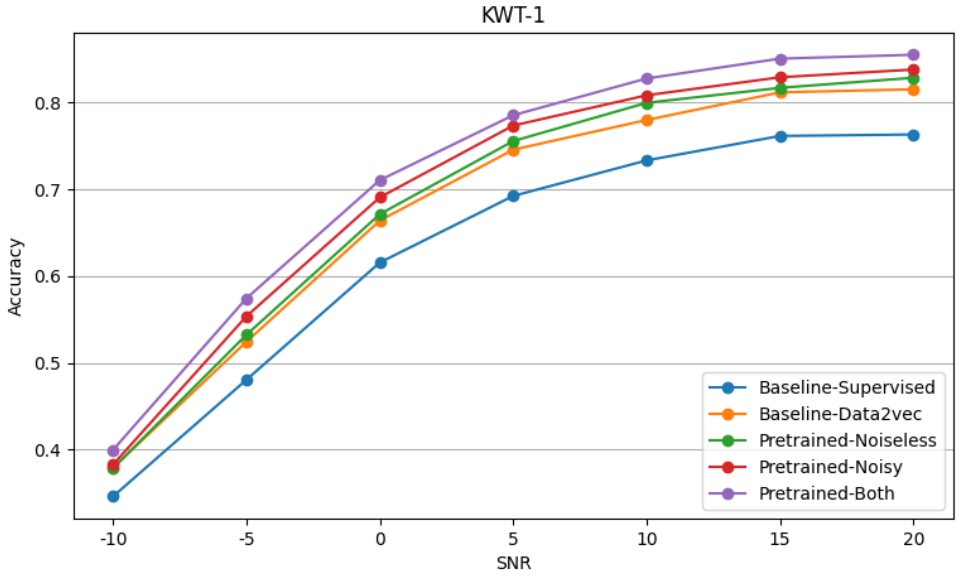

Figure 4.1: Test plot

Below the accuracies for the KWT-2 models are presented in Table 4.3 the exact accuracies can be read and on Figure 4.2 a visual comparition is presented.

KWT-2

SNR	Baseline - Supervised	Baseline - Deta2vec	Pretrained - Noiseless	Pretrained - Noisy	Pretrained - Both
-10	0.3131	0.3983	0.3888	0.4110	0.4483
-5	0.4667	0.5697	0.5728	0.5870	0.6279
0	0.5958	0.7119	0.7050	0.7294	0.7556
5	0.6792	0.7974	0.7908	0.8123	0.8349
10	0.7312	0.8372	0.8320	0.8511	0.8653
15	0.7536	0.8586	0.8476	0.8673	0.8857
20	0.7655	0.8632	0.8576	0.8736	0.8871

Table 4.3: Shows the accuracy of the five models trained using KWT-2, when tested on data with BUS and BBL noise added. The first two columns are the baseline test, a supervised and a SSL trained using only noiseless data. The last three are trained using different data in the pretraning, one uses noiseless, the second uses noisy, and the last uses noisy for the student and noiseless for the teacher.

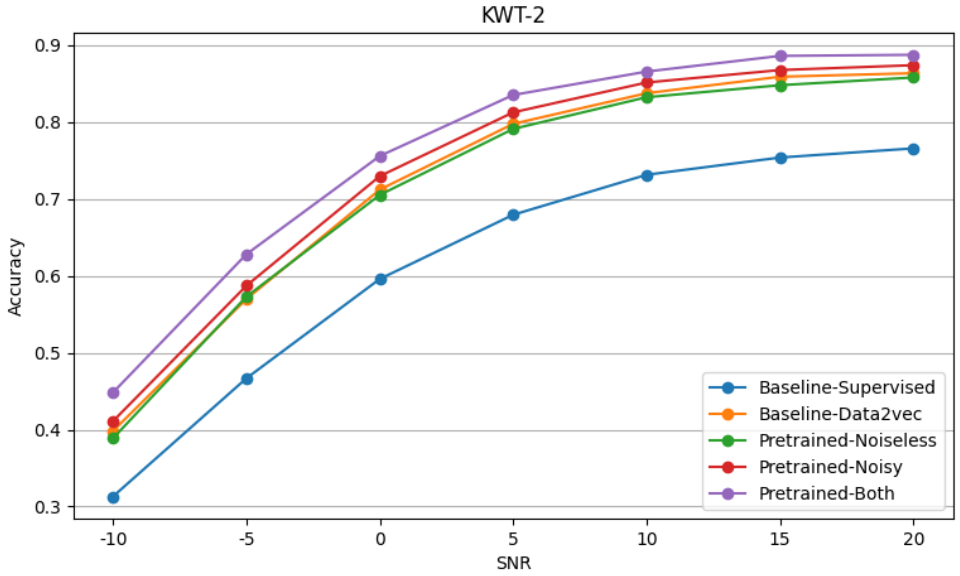

Figure 4.2: Test plot

Lastly the KWT-3 models, on Table 4.4 the exact accuracies can be found and a more visual comparison on Figure 4.3

KWT-3					
SNR	Baseline - Supervised	Baseline - Deta2vec	Pretrained - Noiseless	Pretrained - Noisy	Pretrained - Both
-10	0.2970	0.3847	0.3602	0.3814	0.4068
-5	0.4361	0.5546	0.5083	0.5416	0.5834
0	0.5738	0.6938	0.6472	0.6794	0.7187
5	0.6626	0.7832	0.7253	0.7592	0.7965
10	0.7123	0.8264	0.7695	0.7985	0.8355
15	0.7474	0.8501	0.7950	0.8201	0.8569
20	0.7467	0.8598	0.7953	0.8209	0.8575

Table 4.4: Shows the accuracy of the five models trained using KWT-3, when tested on data with BUS and BBL noise added. The first two columns are the baseline test, a supervised and a SSL trained using only noiseless data. The last three are trained using different data in the pretraning, one uses noiseless, the second uses noisy, and the last uses noisy for the student and noiseless for the teacher.

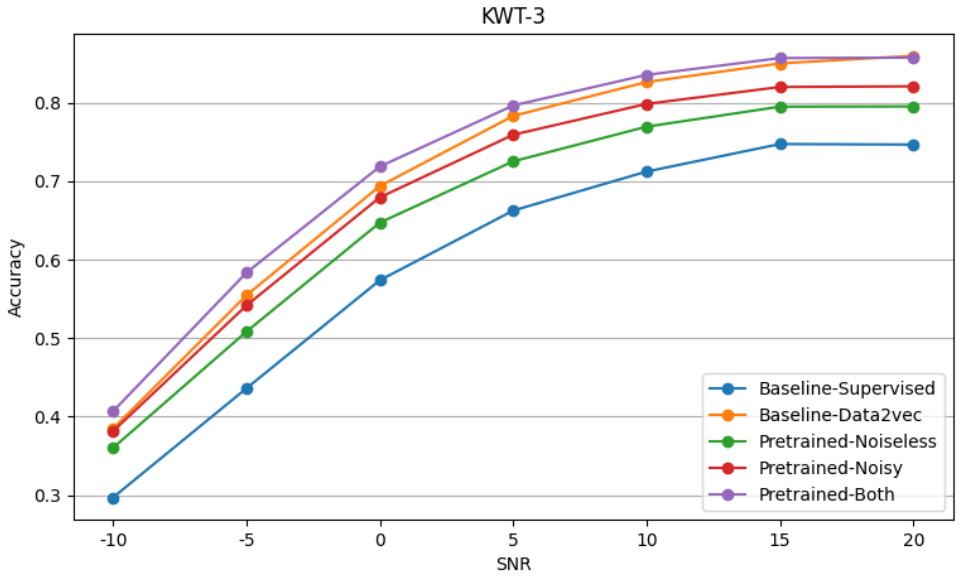

Figure 4.3: Test plot

4.4.2 Testing on BUS, BBL, CAF, and SNN

In this section the accuracy of the trained models when they are tested on new recordings with BUS, BBL, CAF, and SNN noise added. On Table 4.5 the accuracies of the KWT-1 models are presented and on Figure 4.4 the accuracies are plotted for a more visual comparison.

KWT-1

SNR	Baseline - Supervised	Baseline - Deta2vec	Pretrained - Noiseless	Pretrained - Noisy	Pretrained - Both
-10	0.2436	0.2604	0.2541	0.2721	0.2785
-5	0.3965	0.4260	0.4319	0.4583	0.4642
0	0.5443	0.5988	0.6005	0.6304	0.6422
5	0.6393	0.6941	0.7169	0.7378	0.7523
10	0.7189	0.7710	0.7809	0.7916	0.8161
15	0.7544	0.7974	0.8118	0.8234	0.8442
20	0.7706	0.8215	0.8289	0.8420	0.8608

Table 4.5: Shows the accuracy of the five models trained using KWT-1, when tested on data with BUS, BBL, CAF, and SNN noise added. The first two columns are the baseline test, a supervised and a SSL trained using only noiseless data. The last three are trained using different data in the pretraning, one uses noiseless, the second uses noisy, and the last uses noisy for the student and noiseless for the teacher.

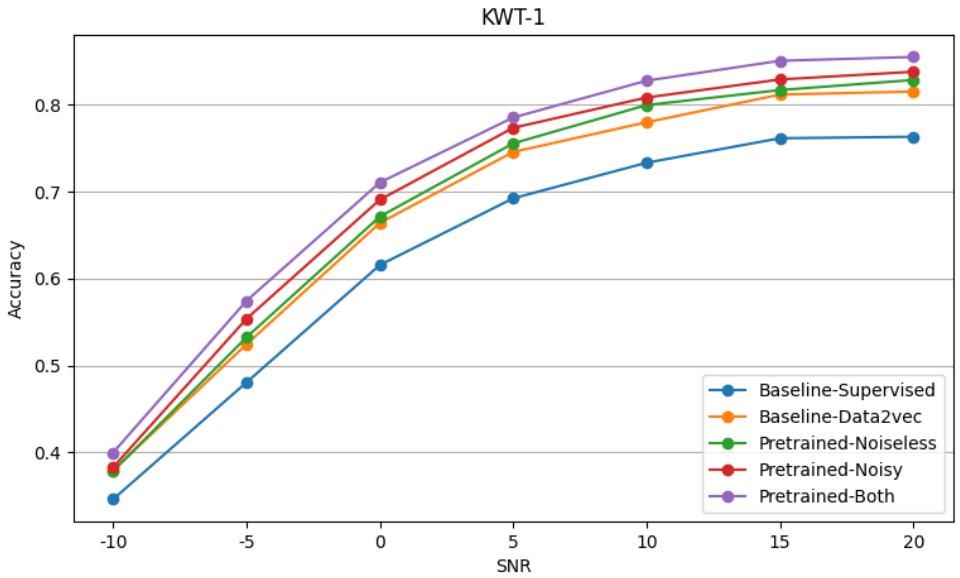

Figure 4.4: Test plot

Below the accuracies for the KWT-2 models are presented in Table 4.6 the exact accuracies can be read and on Figure 4.5 a visual comparition is presented.

KWT-2

SNR	Baseline - Supervised	Baseline - Deta2vec	Pretrained - Noiseless	Pretrained - Noisy	Pretrained - Both
-10	0.2093	0.2704	0.2748	0.2683	0.2999
-5	0.3603	0.4612	0.4612	0.4621	0.5130
0	0.5149	0.6416	0.6393	0.6622	0.6899
5	0.6195	0.7605	0.7484	0.7701	0.7957
10	0.6990	0.8210	0.8158	0.8357	0.8584
15	0.7408	0.8503	0.8458	0.8632	0.8797
20	0.7710	0.8668	0.8568	0.8810	0.8898

Table 4.6: Shows the accuracy of the five models trained using KWT-2, when tested on data with BUS, BBL, CAF, and SNN noise added. The first two columns are the baseline test, a supervised and a SSL trained using only noiseless data. The last three are trained using different data in the pretraning, one uses noiseless, the second uses noisy, and the last uses noisy for the student and noiseless for the teacher.

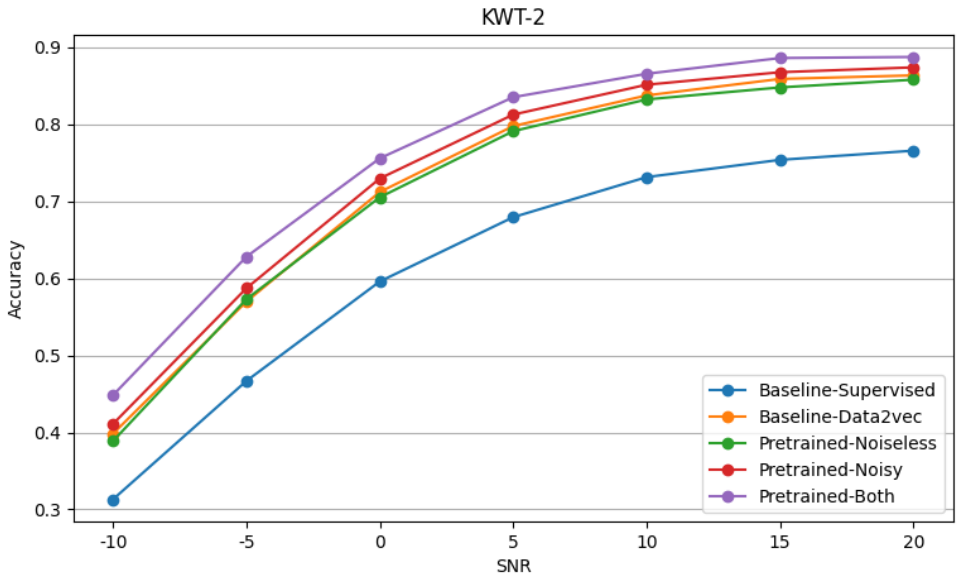

Figure 4.5: Test plot

Lastly the KWT-3 models, on Table 4.7 the exact accuracies can be found and a more visual comparison on Figure 4.6

KWT-3

SNR	Baseline - Supervised	Baseline - Deta2vec	Pretrained - Noiseless	Pretrained - Noisy	Pretrained - Both
-10	0.2030	0.2701	0.2427	0.2579	0.2817
-5	0.3507	0.4580	0.4065	0.4350	0.4727
0	0.4961	0.6326	0.5688	0.6115	0.6602
5	0.6061	0.7437	0.6881	0.7190	0.7587
10	0.6806	0.8101	0.7500	0.7842	0.8179
15	0.7256	0.8380	0.7797	0.8069	0.8483
20	0.7522	0.8532	0.8010	0.8223	0.8635

Table 4.7: Shows the accuracy of the five models trained using KWT-3, when tested on data with BUS, BBL, CAF, and SNN noise added. The first two columns are the baseline test, a supervised and a SSL trained using only noiseless data. The last three are trained using different data in the pretraning, one uses noiseless, the second uses noisy, and the last uses noisy for the student and noiseless for the teacher.

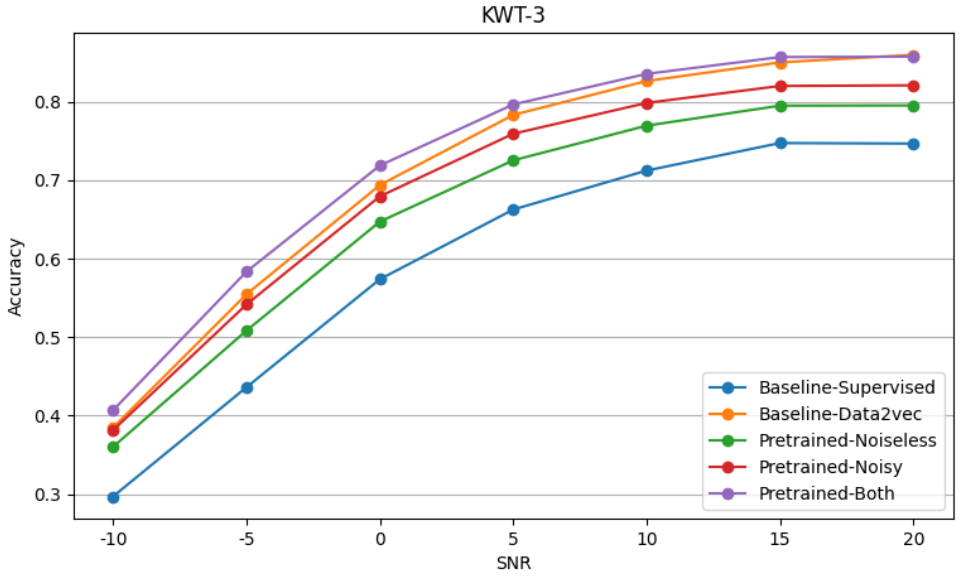

Figure 4.6: Test plot

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*.

Barker, J., Marxer, R., Vincent, E., and Watanabe, S. (2015). The third 'chime'speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 504–511. IEEE.

Berg, A., O'Connor, M., and Cruz, M. T. (2021). Keyword transformer: A self-attention model for keyword spotting. *arXiv preprint arXiv:2104.00769*.

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.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Kolboek, M., Tan, Z.-H., and Jensen, J. (2016). Speech enhancement using long short-term memory based recurrent neural networks for noise robust speaker verification. In *2016 IEEE spoken language technology workshop (SLT)*, pages 305–311. 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.

Loshchilov, I. and Hutter, F. (2018). Fixing weight decay regularization in adam.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

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*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

A | Data

The 35 words included in the Google speech Commands V2 data set.

Label Number	Word	Label Number	Word
0	backward	18	no
1	bed	19	off
2	bird	20	on
3	cat	21	one
4	dog	22	right
5	down	23	seven
6	eight	24	sheila
7	five	25	six
8	follow	26	stop
9	forward	27	three
10	four	28	tree
11	go	29	two
12	happy	30	up
13	house	31	visual
14	learn	32	wow
15	left	33	yes
16	marvin	34	zero
17	nine		

Table A.1: The 35 keywords included in the Google Speech Commands V2 data set.

A.1 Results

		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	0.8420

(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 A.2: The results of training models to a specific noise level with two different types of noise, BUS and BBL. The test consist of the same noises.

		Data2vec pretraining	
snr-10	Baseline	non-noisy	noisy
KWT-1	0.2245	0.2435	0.2306
KWT-2	0.2035	0.2479	0.2346
KWT-3	0.1910	0.2316	0.2504

(a) Snr -10

		Data2vec pretraining	
snr-5	Baseline	non-noisy	noisy
KWT-1	0.3725	0.4006	0.4068
KWT-2	0.3607	0.4119	0.4325
KWT-3	0.3446	0.4075	0.4692

(b) Snr -5

		Data2vec pretraining	
snr0	Baseline	non-noisy	noisy
KWT-1	0.5122	0.6143	0.6363
KWT-2	0.5218	0.6265	0.6500
KWT-3	0.5027	0.5925	0.6639

(c) Snr 0

		Data2vec pretraining	
snr5	Baseline	non-noisy	noisy
KWT-1	0.6473	0.7097	0.7585
KWT-2	0.6334	0.7567	0.7616
KWT-3	0.6139	0.7298	0.7486

(d) Snr 5

		Data2vec pretraining	
snr10	Baseline	non-noisy	noisy
KWT-1	0.7170	0.8084	0.8184
KWT-2	0.7241	0.8282	0.8255
KWT-3	0.6861	0.8051	0.8425

(e) Snr 10

		Data2vec pretraining	
snr15	Baseline	non-noisy	noisy
KWT-1	0.7657	0.8365	0.8473
KWT-2	0.7657	0.8542	0.8688
KWT-3	0.7318	0.8183	0.8343

(f) Snr 15

		Data2vec pretraining	
snr20	Baseline	non-noisy	noisy
KWT-1	0.7998	0.8423	0.8556
KWT-2	0.7949	0.8740	0.8707
KWT-3	0.7816	0.8458	0.8641

(g) Snr 20

Table A.3: The results of training models to a specific noise level with two different types of noise, BUS and BBL. The test consist of four noises, BUS, BBL, CAF, and SSN