A Comparison of the Wavelet Scattering Transform for Feature Extraction Against the CNN in Classification

Johanna Lidholm, jo8208li-s, 990412-2182, LU Faculty of Engineering F19 , johanna.lidholm999@gmail.com
Eliot Montesino Petrén, el6183mo-s, 990618-9130, LU Faculty of Engineering F19 , eliot.mp99@gmail.com
Supervisor: Oskar Keding

I. Introduction

This report investigates the properties of the Wavelet Scattering Transform (WST) as a feature extractor for classification and compares it against the Convolutional Neural Network (CNN) that utilizes feature learning. The second order scattering transform of the WST can be seen as analogous to the initial filter layers of the CNN that both intend to extract features of signals with convolutional operations, where the former is thought to come with some benefits in robustness against noise and scarce datasets, computational cost and "ease of use" reducing the amount of fine tuning parameters. The extracted features are sequentially subject to classification training a fully connected neural network called a dense layer. Specifically, this report focuses on investigating the potential benefits that the WST offers illustrating its' optimal use cases. The comparison between the WST and a multi-layered CNN is made in the context of signal processing, applying the two methodologies to some classification problems in machine learning: a chirp classification toy model XXX, spoken digits classification, and the cocktail party problem on electroencephalography (EEG) data.

II. Theory

This chapter attempts to introduce the convolutional neural network (CNN), in particular the convolutional layers. Then we introduce the Wavelet scattering transform (WST), its constituents, why it is relevant in machine learning models, and lastly, explain the similarities and differences between the CNN and WST for feature extraction.

A. The Convolutional Neural Network

Convolutional neural networks (CNN) are a class of deep neural networks, most commonly known for being applied to images due to its' efficiency in extracting features in higher-dimensional data, such as 2D images. The CNN can however also be applied to 1 dimensional data such as time series signals.

1) General structure of the 1 dimensional CNN: The CNN is structured in two main parts: 1. feature learning, including the convolutional filter layers, and 2. classification in the subsequent fully connected layers. Fully connected layers are also known as dense layers. [14]. Figure 1 shows a general example of a 1D convolutional CNN model illustrating the convolutional layers and its output to the dense layers. The output of the filter layers is known as a latent representation of the input data (represented by the flattened layer) where features of the input has been extracted. The latent representation is not necessarily interpretable for a human in any logical manner but it should be a reduced version of the input, extracting its' features, that can be classified by a simple multi layered perceptron (MLP) consisting of a few dense layers.

2) Feature Learning: A key characteristic of the CNN is that the feature extraction is in fact feature learning, meaning that the filter layers is dependent on trainable weight parameters. This means that the CNN iteratively learns in supervised learning how the data should be represented for effective classification of the labeled input, essentially making it a data driven model. In practice it is thus expected that the CNN requires a lot of data available to be able to learn how the data should be represented as it at the same time also learns how to interpret the latent representation.

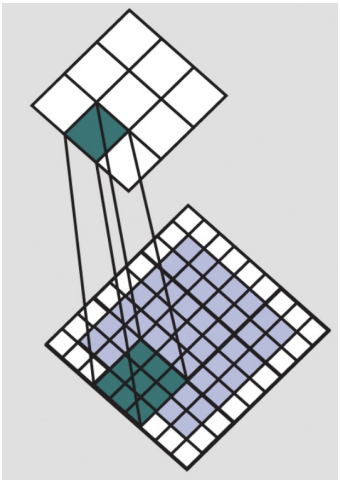

Fig. 2. Illustration of a kernel sweeping over the entire 2D input data, corresponding to a discrete convolution and a pooling operation reducing the input. The input data is the larger image below the $3\,x\,3$ kernel that becomes the reduced output [**kernelsweep**].

The the initial convolutional layers of a CNN performs feature extraction by a series of convolutional operations between the input data, x, and the filter, h, denoted as $x * h$. The general 1D discrete convolution with a filter h is defined in eq 1 [1].

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \tag{1}$$

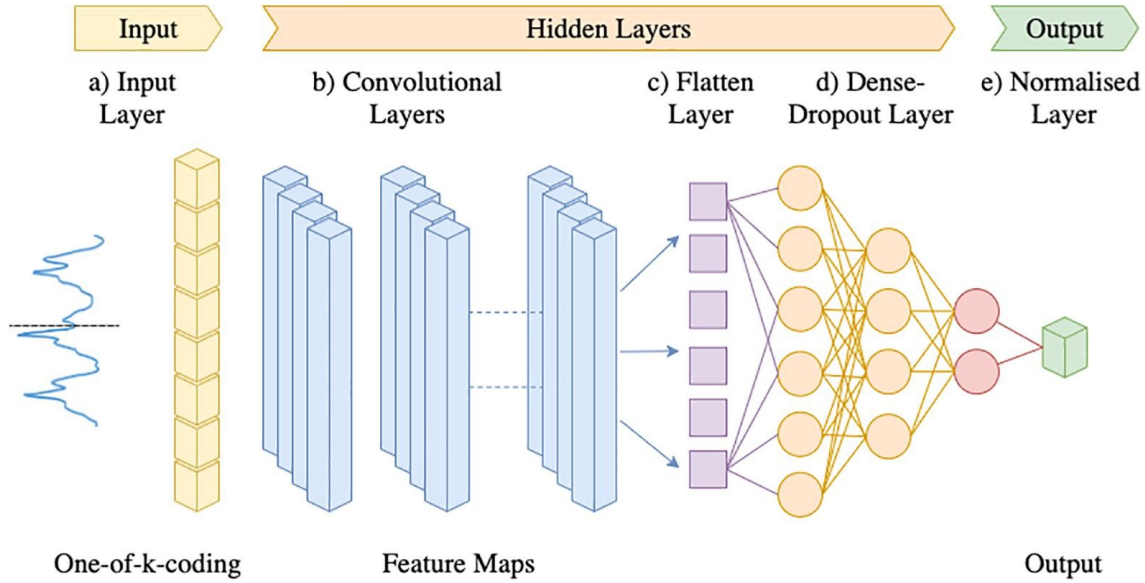

Fig. 1. Example of a multi-layered 1D convolutional CNN. The model takes a 1D signal as input (a) and the input goes through a pipeline consisting of two main parts: convolutional layers in feature learning (b) outputting a latent representation of the input (c) to the dense layers in classification (d) ending in a final guess of the possible classes (e) [10].

The discrete convolution is applied a certain number of times and a kernel sweeps over the entire input reducing the size with a pooling method and "max-pooling" is a common choice choosing the maximum data entry after the operation and omitting the rest. For a lack of a more appropriate visualisation of 1D input, the operation commonly visualized as it is done in figure 2 when the data is 2D. In a 1D convolutional layer the data in the figure would be reduced to 1 dimension.

In a 1D convolutional CNN applied to 1D time series the kernel instead has a dimension of $1 \times l$, where $l \in \mathbb{N}$. The kernel sweeps over the temporal dimension of the time series input.

The chosen output y after pooling is then the input for an activation function, where the rectified linear unit (ReLU) is a common choice enabling the ability to capture nonlinear features of the input data, see eq 2 [2].

$$f(y) = \max(0, y) = \begin{cases} x & \text{if } y > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

The activation function f is thereafter tied to a weight w that is updated iteratively with backpropagation minimizing the error in classification with a given gradient descent method, where a popular choice is the ADAptive Moment estimation (Adam). Dropout is a technique to prevent overfitting by randomly setting weights to 0, so that no weight is overly relied on.

The data driven characteristics of the CNN means in practice that the CNN is a flexible model in the sense it can be adapted to be able to solve various problems of different data inputs (given a suitable model), but it also requires a large amount of data to converge to a solution. Converging to a solution of for the extensive hidden layers means a heavy computational cost and many weight parameters to store after training.

3) Summary: In summary, CNNs are known for its ability to handle various machine learning tasks with high accuracy [7]. They're efficient at sharing information between different layers, which helps them extract important features from input data efficiently [6]. However, using CNNs can be costly in terms of computation, and they tend to have large and complex network structures.

B. Wavelet Scattering Transform

The wavelet scattering transform (WST) is a technique that uses cascades of wavelet transforms, modulus operators and low-pass filters to capture the invariant features and hierarchical structures within a signal (either 1D or 2D). These operations output robust representations that are stable to deformations and noise, enabling a single dense layer to classify the input.

1) The Morlet Wavelet:

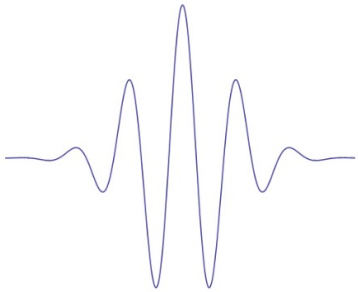

Fig. 3. The real valued Morlet wavelet. The Morlet wavelet could be described as an enveloped sine wave. [**wiki˙morlet**]

A wavelet is essentially an enveloped sinusoid creating a compact and localized waveform in both time and frequency. There are several ways of designing different wavelets but the

standard choice, or at least a common starting point in machine learning (ML) applications, is the Morlet wavelet [13].

Formally, the Morlet wavelet is complex valued and created by an exponential carrier multiplied with a Gaussian window. In this project the real values Morlet wavelet is used, visualized in Figure 3 [16].

2) The Continous Wavelet Transform: In 1946 the physicist Dennis Gabor extrapolated ideas from quantum physics to introduce the use of Morlet wavelets for time-frequency decomposition. Gabor reffered to these waveforms as "atoms" and intended to create a tool for an efficient trade-off between spatial and frequency resolution [11]. The Morlet wavelets was utilized to introduce a type of short-time Fourier transform named after him, the Gabor transform [16].

Gabor showed that time-frequency atoms have minimal spread in the time-frequency plane and that decompositions using wavelets such as the Morlet wavelet is closely related to human perception (such as hearing) [11]. This makes applications using wavelets especially interesting in the field of artificial intelligence.

The continous wavelet transform (CWT) was first introduced in 1984 when Jean Morlet modified the Gabor transform to keep the same wavelet shape over equal octave intervals [16]. The CWT is a convolution of the input data sequence with a set of functions generated by the mother wavelet. Choosing a real valued morlet wavelet as the mother wavelet produces real valued outputs [15].

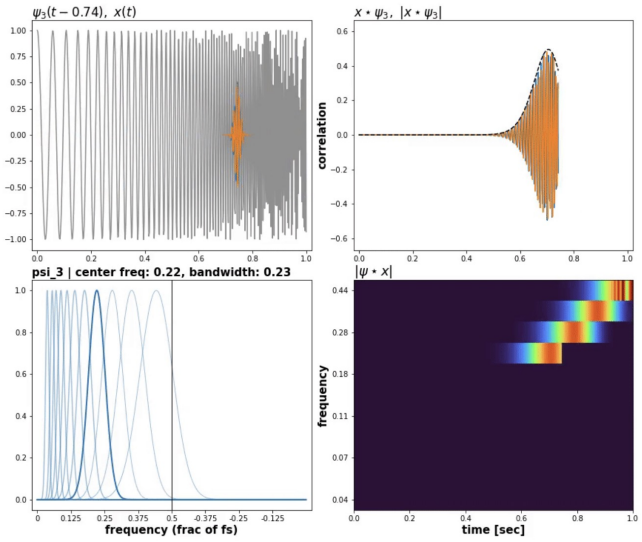

Fig. 4. Still picture of a gif visualizing the CWT [12].

The CWT involves convoluting several wavelets of different scales and frequencies, visually corresponding to stretched or compressed wavelets, against a signal. This provides an output over the temporal dimension when the wavelet is *similar* to the signal it is convolved against. The convolutional operation is perhaps best visualized in video [12] but a picture is provided in figure 4.

The CWT can intuitively be compared with the Fourier transform but with the difference that we get a time-frequency spectrum from CWT whereas the Fourier transform is only present in the frequency domain.

3) The Wavelet Scattering Transform: The WST is a scattering transform computed in an iterative fashion up to the second order and the algorithm is visualized as a tree view in figure 5. The WST includes repetitions of three stages of data processing summarized in figure 6. The transform yields representations of the input signals that are time-shift invariant, robust to noise, and stable against time-warping deformations ideal for ML applications [12]. The WST can be applied on 1D and 2D data, but for the purposes of this project only 1D will be considered [4].

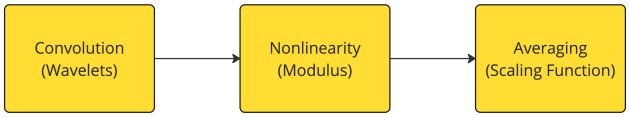

Fig. 6. Every order of the scattering coefficients are computed in main three stages of data processing in the wavelet scattering transform. [4]

A wavelet function (Morlet) is introduced $\psi_{J,\lambda}$ and a scaling function ϕ_J that is a low-pass filter, where J is the number of scales, a pre-defined parameter, and λ is the center frequency of a single wavelet convolution in a CWT. First the zeroth order scattering coefficients $S[0]$ are computed of the input signal x by applying the low-pass filter on the data entries $S_0x = x * \phi_J$ and this is represented by the first row of the single black node in figure 5. The first order coefficients does not carry useful information and are in practice omitted later on in ML applications [9] [4].

The first order coefficients, the output of the first order scattering transform, are obtained by performing the steps in figure 6. The first step is the CWT of the input x for a cascade of wavelets from a filter bank $\psi_{\lambda_1}, \psi_{\lambda_2}, ...$ and the second step is performed computing the absolute value (modulus) $|x * \psi_{\lambda_1}|$ yielding the white nodes in the second row in figure 5. The third step is applied averaging the outputs again with the low-pass filter finally yielding the first order coefficients $S[1]$ $Sx(t, \lambda_1) = |x * \psi_{\lambda_1}| * \phi_J$ represented by the black nodes of the second row in figure 5. This output is saved to be used later in classification.

The second order scattering coefficients are obtained by repeating the three steps in figure 6 for every white node of the second row in figure 5 yielding $S[2]$ $Sx(t, \lambda_1, \lambda_2) = ||x * \psi_{\lambda_1}| * \psi_{\lambda_2}| * \phi_J$ represented by the third row of black nodes in the figure and the second order scattering coefficients are also saved. The WST generally never further computes scattering coefficients of higher orders since every iteration means a significance loss in energy [9]. A vague analogy to this in physics could be the redundancy and loss of information in taking time derivatives of higher orders than 3, "acceleration of acceleration...".

The low-pass filter is smoothing in the temporal dimension enforcing time-shift invariance and robustness against time-warping deformations which are characteristics both useful in

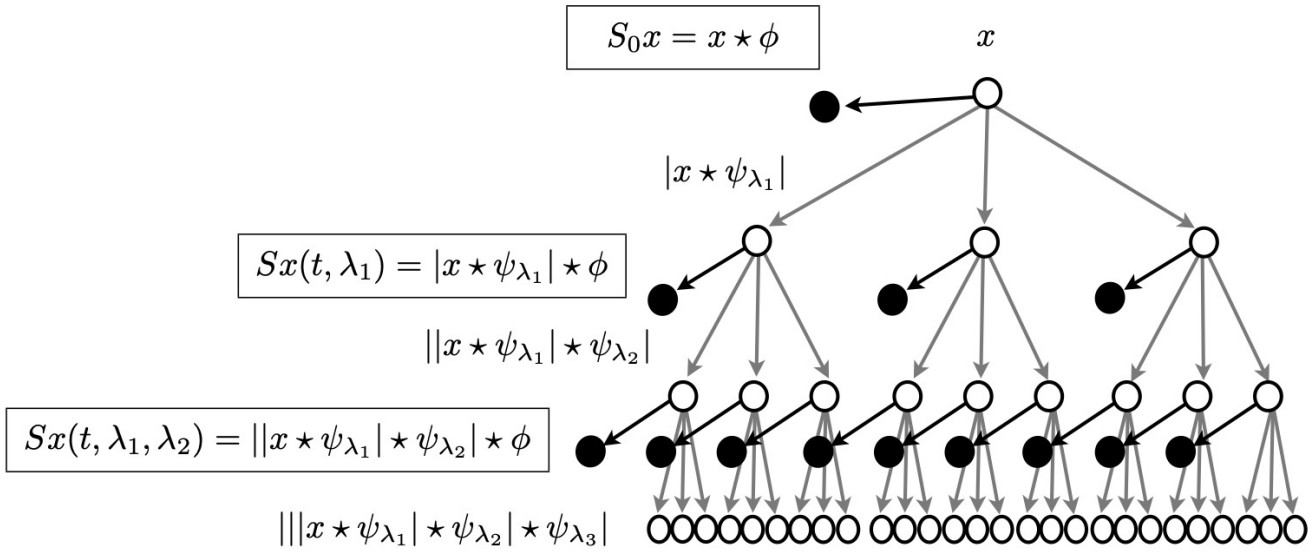

Fig. 5. A scattering transform iterates on wavelet modulus operators $|x * \phi_\lambda|$ to compute cascades of wavelet convolutions and stores the output of low pass averaged scattering coefficients $S_m x$ of order m. [5]

classification, but the operation loses time localization. The purpose of the second order is that it recovers some of this lost information since it is performed on the un-smoothed coefficients represented by the white nodes of the second row in figure 5. The modulus operation is non-linear and one of the purposes of it is that it enables higher order scattering coefficients. Without modulus the second order scattering coefficients would be stacked convolutions, which are equivalent to a single convolution [12].

Time-shift invariance is useful in ML classification since it is primarily interested in "what" is present in data input and not fine aspects of "when" or "how". Time-warping deformation refers to modifying a signal by stretching or compressing its time axis. This kind of deformation can affect the temporal relationships within the signal without necessarily altering its frequency content significantly. Warp stability is thus useful since it is stability against small changes in the rate of the source process, while the source process remains the same. Examples are the variance of rate of speed in speech or the doppler effect, which does not change the label of the input. Words keep the same meaning invariant of speed and an ambulance is still an ambulance invariant of direction [12].

The characteristics of the WST explained above is described as that it derives low-variance features of signals. The feature extraction of the input is stable and does not change dramatically with small changes in the input data, thus making them a reliable tool for training machine learning models in classification where generalization is of high interest. The coefficients are a compact representation of the input signal, but abstract for a human, very similar to the goal of the filter layers of a CNN.

4) WST and CNN analogy and differences: Assume that the first and second order scattering coefficients of the WST are input to a simple MLP consisting of a single dense layer. This WST neural network model can be seen as an analog equivalent to a CNN that has two convolutional layers.

How does the WST and the analog CNN model differ? The feature extraction of the CNN, the feature learning, requires traing while the feature extraction of the WST are only dependent on the fine tuning parameters number of scales J and number of wavelets per octave Q. The WST nueral network model (from now on called simply the "WST model") only trains in the single dense layer. In summary this means that the approach of the CNN model is generally more data driven than the approach of the WST model. In theory this should translate to that a CNN model equivalent to the WST model requires more data to be effective as it has more trainable parameters in both how the features should be extracted and how the output should be classified. The WST model is thus thought to be more effective when few data are available but it could also provide significant benefits in robustness against corrupted data and disturbances since it should derive low variance features of the input efficiently. If the WST model is able to perform at least as accurately as the CNN model, then it has done so for a lower computational cost and it does not have to store as many parameters.

III. Chirp Classification

This part of the project aims to explore the basic concepts of the WST and compare it to the performance of a convolutional neural network using generated chrip data. Two main ideas are explored. First: How does a model using the WST of the input data compare to a CNN in terms of the amount of training data provided for uncorrupted signals? Secondly: Given a constant amount of training data, how does a WST compare to a CNN when the signal is corrupted for different noise levels?

The WST incorporates as many convolutions as the number of scattering coefficients employed. When configuring a CNN for comparison with the WST network, we consequently utilize an equivalent number of 1D convolutional layers as scattering coefficients. The quantity of wavelets utilized in the WST transform aligns with the number of filters in the 1D convolutional layers of the CNN. Although the CNN necessitates more parameters for tuning compared to the WST network, it operates with the original signal as input. In this section, we delve into the performance analysis between these two model selections.

A. WST and CNN equivalent models

The WST contains the same amount of convolutions as the number of scattering coefficients used. When setting up a CNN to compare the WST network with we thus have the same amount of 1D convolutional layers as scattering coefficients. The number of wavelets used in the WST transform is set equal to the number of filters in the 1D convolutional layers in the CNN. The CNN will have more parameters to tune than the WST network but will use the original signal as input. In this section, we study the performance between these two model choices.

B. Input data

The input data consisted of randomly generated chirp signals, with an amplitude range between 0.5-1.5, a frequency range between 10-100 Hz, a duration of 1s, and a sampling rate of 1000 Hz. The data was generated in pairs, one with increasing frequency over time and one with decreasing frequency over time. An illustration can be seen in Figure 19. The classification problem is to let a WST model and CNN classify whether the frequency is decreasing or increasing for a given chirp signal.

The generated signal also had an additional noise parameter, corrupting the signal with Gaussian noise. Figure 20 shows a few examples of how a corrupted signal can look for different noise levels.

C. Model structures

1) WST model: The WST model was constructed by transforming the input data through a scattering transform where the first and second order were kept. The transform was then normalized with the mean and standard deviation. Finally it was passed as input to a single-layer neural network with LogSoftMax activation function, see Figure 7. To simplify the examination, a few parameters where set constant during the process. J, which represents the number of scales and Q which represents scale per octave summarized in table I. For more details, see the Kymatio documentation [9].

TABLE I
Summary of Parameters for Wavelet Scattering Transform Chirp Classification

Parameter	Value
Activation Function	LogSoftMax
Number of Scales (J)	10
Scales per Octave (Q)	8

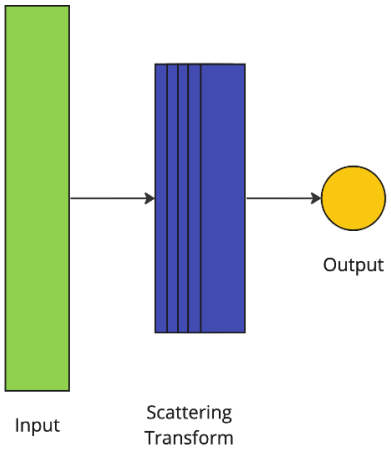

Fig. 7. CNN model with WST as input used for chirp classification, consisting of one scattering transform and a single dense output layer.

2) CNN model: The CNN model consisted of two convolutional 1D layers with the same number of filters as the number of wavelets in the ReLu-activation functions were used in the convolutional layers and with $kernel\ size = 3$. Max pooling was added after the convolutional layers with $pool\ size = 2$. After flattening, a dropout of 50% was added. The last layer was a dense layer with the sigmoid activation function. An illustration of the architecture can be found in Figure 8.

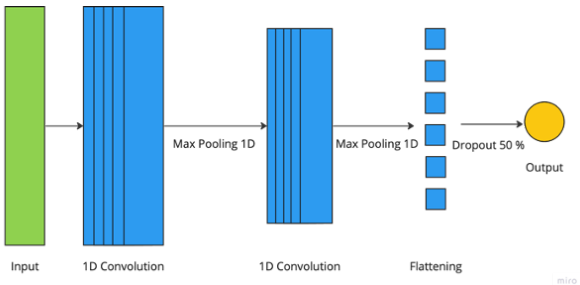

Fig. 8. CNN used for chirp classification, consisting of two convolutional layers with Max-pooling and 50% dropout.

D. Simulation setup

In the first simulation, the parameters were initialized, and the number of chirp pairs used in the training data was varied from 2 to 20. Each pair consisted of both increasing and decreasing chirps, ensuring an equal representation of different

signal patterns. Additionally, a fixed number of chirp pairs were generated for test data evaluation.

The code iterated through different numbers of chirp pairs for training, generating random samples for each iteration. To mitigate the effects of random variations, each iteration was repeated 10 times. Subsequently, both the WST and CNN models were trained on the same training data and evaluated on the same test data.

Given the difference in model complexity, the number of epochs for training was set differently for the CNN and WST. Specifically, the CNN was trained for 30 epochs, while the WST was trained for 5 epochs. This adjustment was determined through empirical testing, aiming to balance model performance without overfitting. Figure 9 shows a flowchart of the simulation process.

In the second simulation, the procedure was similar, but instead of varying the number of chirp pairs, different noise levels were introduced to the signals. The number of chirp pairs in the training data remained constant at 12 pairs. This simulation aimed to assess the robustness of both models to varying levels of noise in the input signals.

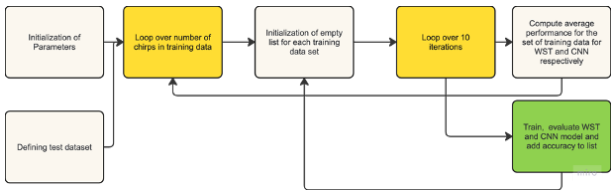

Fig. 9. Illustration of the simulation setup for evaluating performance on different amounts of training data for the WST and CNN.

E. Results

Figure 10 illustrates the classification accuracy of both the WST and CNN models under varying quantities of chirp pairs in the training data, with evaluation conducted on a consistent set of 30 chirp pairs in the test data. Notably, the WST exhibits marginally superior performance when trained on smaller datasets. However, as the volume of training data increases, the CNN outperforms the WST, demonstrating its capability to effectively handle larger datasets.

Figure 11 presents the outcomes obtained by varying noise levels in the input signals. The CNN demonstrates superior performance at lower noise levels. Conversely, as the noise levels escalate, the WST showcases efficacy to maintain higher accuracy levels, thereby indicating resilience to distortions.

F. Discussion

Our investigation of the scattering transform aimed to conduct a comprehensive comparison between models based on Convolutional Neural Networks (CNN) and the Wavelet Scattering Transform (WST). Our findings reveal that the WST offers a compelling advantage by effectively reducing model parameters while retaining crucial features from the input data. Remarkably, achieving high accuracy with the WST necessitated fewer epochs, particularly evident in uncorrupted signal scenarios. Furthermore, when confronted with

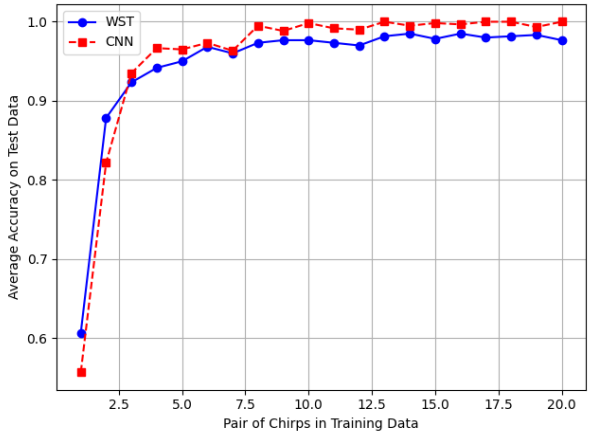

Fig. 10. Test performance of the WST and CNN model. WST performs slightly better for smaller training datasets while CNN is superior when the training dataset is large. Hyperparameters used are: $epochs\_cnn = 50$, $epochs\_wst = 5$, $batch\_size = 1$.

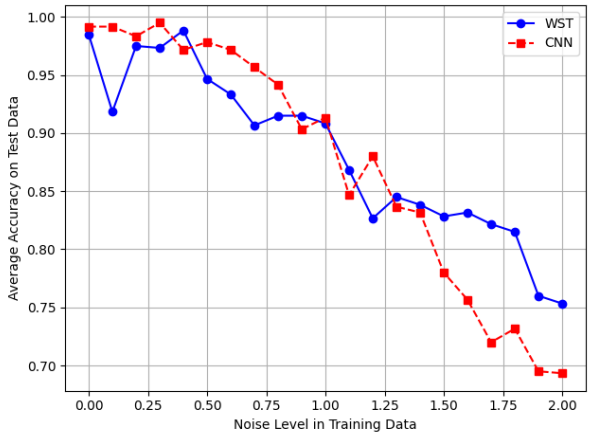

Fig. 11. Test performance of the WST and CNN model with variable noise levels. CNN overall performs better for lower noise levels. Hyperparameters used: $epochs\_cnn = 50$, $epochs\_wst = 5$, $batch\_size = 1$.

limited training datasets, the WST consistently outperformed the CNN, showcasing its robustness and efficiency.

The true strength of the WST emerged when subjected to signal corruption through the addition of noise. Even in the presence of high noise levels, the WST demonstrated proficiency in pattern extraction, underscoring its resilience to noise-induced distortions. Consequently, the WST emerges as a compelling option for feature extraction, especially in scenarios involving limited training data, high noise levels, or where the computational burden of training with a CNN becomes prohibitive.

IV. Spoken Digits Classification

In this section, we conduct an analysis similar to the one found in the previous chapter, contrasting the WST and CNN methodologies. However, our focus shifts to a different challenge within signal processing and machine learning: spoken digits classification. Conducting a similar analysis on a WST model against an equivalent CNN model introduced challenges in this application that is based on real world data.

In spoken digits classification, an artificial neural network model should be able to classify which digit is spoken from an audio recording. The data, the audio recordings, are measurements from the Free Spoken Digits Dataset (FSDD) [8]. Spoken digits classification could be said to be a 1D signal processing equivalent to the classic machine learning problem of written digits classification, commonly applied on data from the MNIST database providing images (2D) as input for training CNN's [3].

Tests were conducted comparing the WST model to an equivalent CNN model but the CNN model had to be altered due to limitations in model structures and a modified CNN model was introduced instead (see subchapter B). Two tests were conducted, 1: Training and validating the two models varying the size of the training dataset. 2: Training and validating the two models increasing the amount of (Gaussian) noise, given a constant (and sufficient) size of the training set.

A. Input data

The data available is from the FSDD [8] and it contains audio recordings of 6 English speakers of a total of 3 000 recordings of single digits pronounced in isolation, meaning that the recordings start and ends with silence. Hence, there are 50 recordings of every digit per speaker. The sample rate of the audio recording are 8 kHz.

The input data was subject to some minor preprocessing. The data was normalized, squashing the amplitude of the time series into an interval of $[-1, 1]$ and this is a common step in training deep neural networks to improve weight updates in training and numerical stability. All sample time series's were either truncated or zero-padded to set to a common length of $2^{13} = 8196$ corresponding to slightly above 1 second. Some samples after preprocessing are visualized in figure 21.

A test dataset was set aside covering 10% of all available data in total of 300 samples. The test dataset made sure to have an equal distribution of classes, meaning that it was the same number of samples of every spoken digit recording of 30 digits each.

1) Test of varying sizes of training sets: One of the tests included varying the amount of available data in training. A total of 10 training sets were made covering from 10% to 100% of the complete training dataset (2 700 samples). The sizes of the test cases are visualized in figure 12 and a full summary of training sizes is included in table III

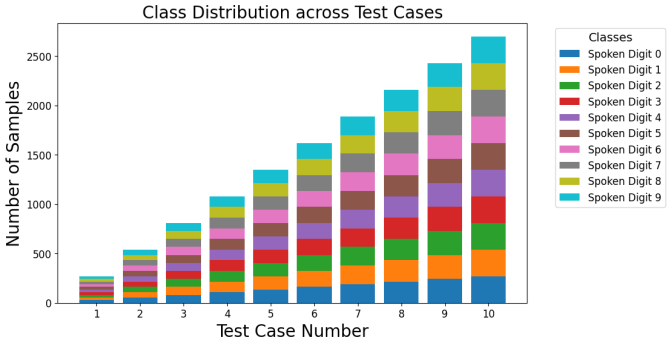

Fig. 12. Equal class distribution across 10 test cases. Total samples in training vary from 270 to 2700. Training sets of every test case are super-sets of the previous ones.

TABLE II
Test Cases with Varying Sample Sizes

Test Case	Training set availability	Samples per Digit	
		Training Set	Test Set
1	100%	270	30
2	90%	243	30
3	80%	216	30
4	70%	189	30
5	60%	162	30
6	50%	135	30
7	40%	108	30
8	30%	81	30
9	20%	54	30
10	10%	27	30

With the objective to increase the quality of the test, variance in training was decreased and subsequently the results as well by making every training set a super-set of the smaller ones. Every incremental larger set were made sure to contain the same samples as far as it was possible from the previous sets.

2) Test of varying corrupting noise: The second test in the analysis involved corrupting the data by adding Gaussian noise across the entire sample. For every test set Gaussian noise was simply added to both the (uncorrupted) training set and the test set. The different test cases are summarized in table III and some corrupted samples are previewed in figure 22 of noise level $\sigma = 0.4$.

TABLE III
Test Cases with Varying Corrupting Gaussian Noise

Test Case	Noise level (standard deviation)
1	0.0
2	0.11
3	0.22
4	0.33
5	0.44
6	0.56
7	0.67
8	0.78
9	0.89
10	1.0

B. Model structures

The CNN model that is analog to the WST based model requires to have the same amount of filters in the 1D convolutional layers as the number of scattering coefficients of the WST in order to satisfy equivalence. This requirement makes it practically not feasible to define a CNN model that is equivalent to the WST model, because the WST outputs too many scattering coefficients.

The WST outputs a number of scattering coeffients equal to the number of entries in the input data (maximum 2 700 recordings) and it is in practice not possible, nor desirable, to define a CNN model with this many filters. The number of filters are in fact absurdly large and thus the Python package Tensorflow (a high level machine learning platform) used in implementation does not accept that many filters and raises an error. Modeling a CNN with 2 700 filters is especially absurd if one would consider modeling a CNN facing the spoken digits classification problem for the first time invariant of the WST model. Two 1D convolutional layers with this many filters would in practice mean an extreme amount of weights, computational cost in training and an unnecessary size of the network and complexity, thus a modified CNN model is defined in subchapter IV-B2.

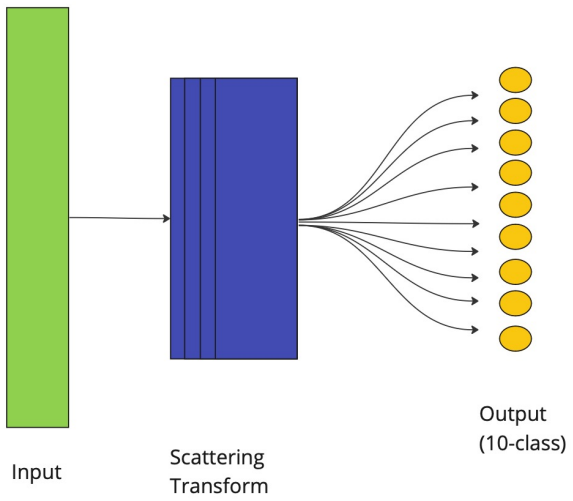

Fig. 13.

TABLE IV
SUMMARY OF PARAMETERS FOR WAVELET SCATTERING TRANSFORM SPOKEN DIGITS CLASSIFICATION

Parameter	Value
Activation Function	LogSoftMax
Number of Scales (J)	8
Scales per Octave (Q)	12

1) WST model: The WST model was constructed in the same manner as it was in chirp classification adjusting for a 10-class classification instead of a binary classification, see figure 13. Especially note that LogSoftMax also used previously is able to handle multiclass classification. The number of scattering coefficients from the scattering transform are dependent on the size of the input data. Parameters are summarized in table IV

2) Modified CNN model: The obstacle of not being able to define a CNN model equivalent to the WST model is substantial and it is in itself a result worthy of discussion in coming chapters but in order to proceed with the analysis the problem was addressed by trying to design a CNN model that stand on somewhat equal grounds to the WST model by creating a new CNN model that takes inspiration from the WST equivalent CNN model. The WST equivalent CNN model had too few filters but complexity can be added in other ways than simply adding additional filters.

Starting from the equivalent CNN model with the objective of maximizing test accuracy in the new CNN model general CNN modeling tactics were applied expanding it's complexity and ability to extract features of the input data. Finally the "modified CNN model" was defined, please see figure 14. The modified CNN model compared to the CNN model in the previous chapter (figure 8) has a third 1D convolutional layer, one added dense layer and Softmax activation in the last dense layer for 10-class classification. Adding an additional 1D convolutional layer expands the model's ability in feature extraction and increasing the number of filters for every layer is done so that the model can find more fine features for every convolution. The added dense layer enhances the model's ability to interpret the extracted features and classify the input. Both these additions increased the accuracy in classification of the CNN model approaching the performance of the WST model.

Adding a third 1D convolutional layer would WST-terms correspond to applying a third scattering transform yielding

C. Simulation setup

Two simulations ran corresponding to each test and the simulation seen as a flowchart in figure 15.

In both simulations data is fetched from the FSDD, pre-processed and split into test and training set. In the varying training sets test an extra step is added where indices of the maximum size training set are randomly selected with an equal distribution of classes. Afterwards 10 sets of indices of the training sets are created where every set are super-sets, explained in chapter IV-A1. Parameters are initialized and the simulation loop is ready to run.

In both tests the code iterates through 10 test cases and to mitigate the effects of random variations, each iteration was repeated 3 times. Further iterations would take too much time due to a substantial computational cost in every iteration. Subsequently, both the WST and the modified CNN model were trained on the same training data and evaluated on the same test data in every test case.

Given the difference in model complexity, the number of epochs for training was set differently for the CNN and WST. Specifically, the CNN was trained for 15 epochs, while the WST was trained for 10 epochs. This adjustment was determined through empirical testing, aiming to balance model performance without overfitting.

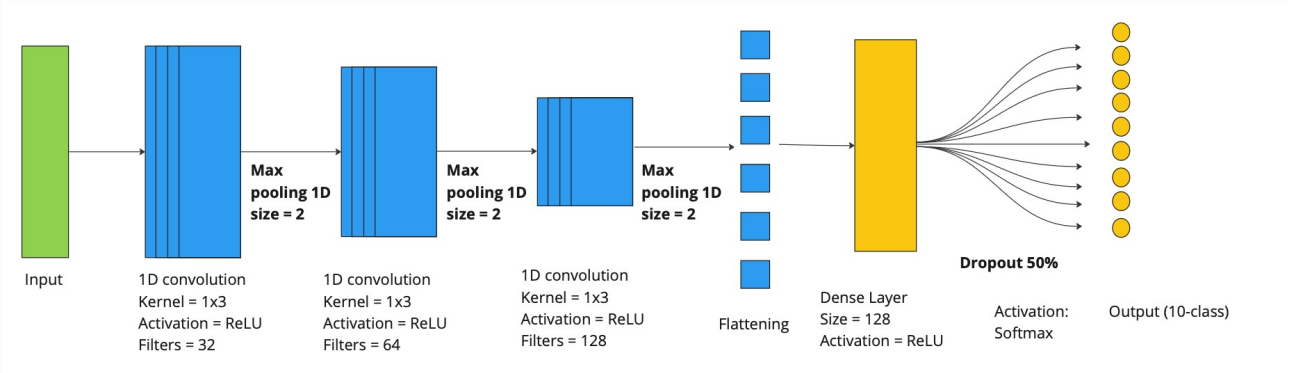

Fig. 14. The modified CNN model that is similar to the WST model but not strictly equivalent. Compared to the equivalent CNN model this one has fewer filters but one added 1D convolutional layer and one added dense layer to compensate for the few filters.

In the second simulation, the procedure was similar, but the preparation of indices to omit from training is skipped and instead a new step is added for every test case in the simulation loop where Gaussian noise is added for a given level to the uncorrupted signal. Gaussian noise is not added in the first simulation where training sets are varied in size. The size of the training set was set to a constant in the second simulation to the maximum size of the training set covering 100% the training set corresponding to 90% of all data available from the FSDD.

D. Results

Figure 16 illustrates the classification accuracy on the test dataset of both the WST and the CNN model under varying availability of training data (total samples of voice recordings of spoken digits). The evaluation was conducted on a consistent test set of 300 samples (10% of total dataset). Both the training sets and the test set had an equal distribution of classes. The total running times of the two tests are long taking around 2 hours, see table V.

The WST model outperformed the CNN model in all tests and especially retains accuracy more effectively in smaller datasets, but there is still a loss of accuracy for smaller training sets in both cases.

Figure 17 contains the results of the test that adds Gaussian noise of varying levels in the input signals. The WST model outperforms the CNN model for all noise levels and the difference in accuracy is somewhat constant in quantitative terms. Both models show exponential loss of accuracy for increasing noise levels. At noise level 1 the accuracy of the CNN corresponds to pure guesses (10%). The accuracy of the WST model at this level is above 10% indicating it is still able to extract features from the input data.

TABLE V
SUMMARY OF APPROXIMATE RUNNING TIMES FOR SIMULATIONS
Using Macbook Air 2020 model

Simulation	Running Time (hours)
Varying training set sizes	2.0
Varying corrupting noise	2.5

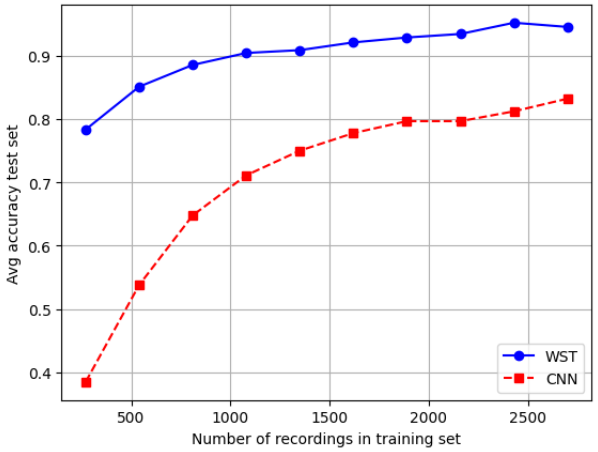

Fig. 16. Test performance of the WST and the (modified) CNN model. The WST model outperforms the CNN model in all tests and especially retains accuracy more effectively in smaller datasets. Hyperparameters used are: $epochs\_cnn = 15$, $epochs\_wst = 10$, $batch\_size = 32$.

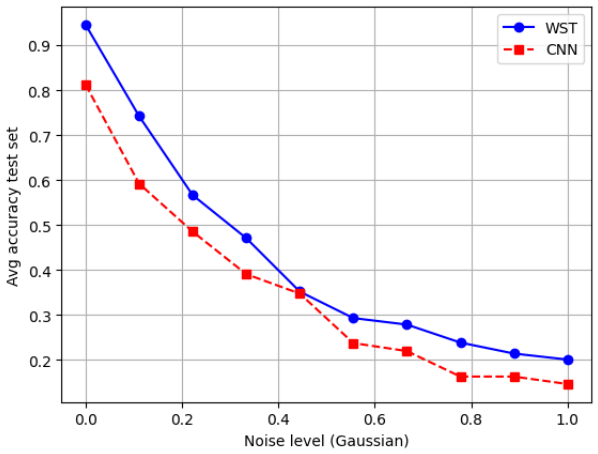

Fig. 17. Test performance of the WST and the (modified) CNN model. The WST model outperforms the CNN model overall and the difference, in quantitative terms, between the two curves are almost constant. Hyperparameters used are: $epochs\_cnn = 15$, $epochs\_wst = 10$, $batch\_size = 32$.

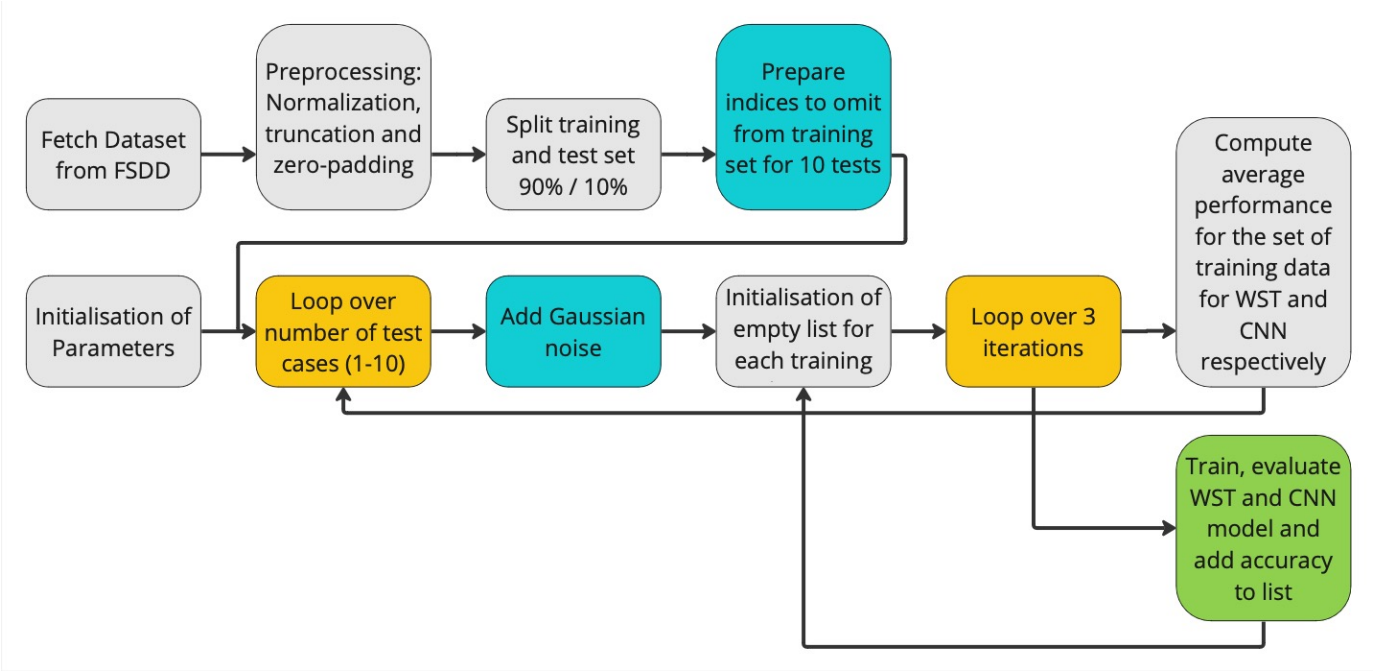

Fig. 15. Illustration of the simulation setups for evaluating performance on varying sizes in training sets and varying corrupting noise for the WST and the modified CNN. The turquoise boxes are optional for said tests. "Prepare indices to omit..." is active in the test of varying test sets and the other is inactive. In the test of varying corrupting noise "Add Gaussian noise" is active and the other inactive.

E. Discussion

This chapter included two tests that aimed to compare two models adopting two different methodologies based on Convolutional Neural Networks (CNN) and the Wavelet Scattering Transform (WST) applying it on measured voice data in the spoken digits classification problem.

The results of the performed tests show (again) that the WST is able to reduce parameters that are subject to fine tuning in modeling (J and Q) while still performing accurate classification, even outperforming the modified CNN model in all tests in spoken digit classification. The most accurate classification can be identified from the WST model at full training set size and with no corruption in data at satisfactory 95% accuracy compared to the CNN model at slightly above 80% accuracy. The WST model did also in this application converge in training earlier than the CNN model in 10 epochs compared to 15 epochs in the CNN model, but the difference is not as large as it was for Chirp Classification. Setting 15 epochs in the WST model would possibly cause overfitting.

Two trends were implied qualitatively from the results in Chirp Classification: the WST model is more robust against noise and it is more efficient in feature extraction when few samples are available. How does these trends compare to the results of the spoken digits classification? The results in the first test imply that the WST model is robust in feature extraction when data is scarce, increasing in robustness for smaller datasets. The WST model is retaining slightly below 80% accuracy when 27 samples per digit is available compared to slightly below 40% accuracy in the CNN model falling off exponentially. The CNN model parameters are constant however and not dynamic in the number of filters as it was for chirp classification where the number of filters were set equal to the number of coefficients output from the second order scattering transform dependent on size of input data. This might affect results if the CNN is overfitting at the lower levels of available data in the test, since it was initially modeled for the maximum training set size. The number of filters applied for small training sets is still relatively small compared to the WST model that has about 270 spectral coefficients output at this level from the second order scattering transform.

In the second test the results are not symmetrical to the ones in Chirp Classification since the slope of the graphs of the WST and modified CNN models seem to be about the same for all noise levels and there is seemingly a constant difference between the two in every test case. The WST outperforms the CNN likely because it is evidently more efficient in feature extraction in this problem overall and is not due to robustness against noise. One could draw the conclusion rather that the WST model does not introduce weakness to noise compared to the alternative. On the more speculative side, one could argue that the outperforming WST model is enhanced becuase of inherent measurement noise or process noise since the data is not generated but measured. The recordings are however not very noise judging from just looking at the data and listening to the audio files.

It must also be stated, since it admittedly is a significant result as well, that aside from fewer fine tuning parameters the complexity of the WST model is significantly lower in the size of the network (number of weight parameters) and the computational cost is significantly lower in the WST model as well. The reason for this is due to the nature of CNN's where

the feature extraction of the CNN model, the convolutional layers, are in fact a deep network itself with weights that are subject to training and stores many weight parameters. The feature extraction in the WST model on the other hand is ”automatic”, it performs the same mathematical operations every time it is applied and evidently it is still able to extract features that are sufficiently clear for a single dense layer to learn to classify the 10 spoken digits. This is also the reason why the WST is more accurate when data is scarce, it does not learn how features should be extracted and there are fewer weights to train. For future work, the computational cost in training the CNN in the simulation loop could have been reduced by introducing a pretrained CNN as a starting point for training.

How is the overall outperforming WST model explained? One could speculate, based on the WST and CNN analogy, that the WST is able to have many more equivalently ”filters” while still being computeable in a reasonable timeframe whereas the CNN model, due to it’s deep learning architecture in feature extraction, is not computeable since it increases in complexity too much. This enables for robust feature extraction and a single dense layer at the end of the pipeline is enough to classify. The CNN outperformed the WST in Chirp Classification when more data was available, it is a data driven model, but when there is a lot of data then the WST equivalent CNN model is not feasible anymore and fails to provide as efficient feature extraction. In a real world scenario applying a CNN model however, one would probably not start with the WST equivalent CNN model but would perhaps start with more common options such as robust preprocessing options such as a time-frequency transform providing a spectrogram as input to a CNN model, essentially also providing automatic feature extraction before being subject to deep feature learning.

The WST is evidently a compelling option for feature extraction when applying it to a real world scenario containing voice data, especially in scenarios involving limited training data. The computational burden of training a CNN becomes more apparent in real world scenarios where training sets are generally larger, especially in a multi-class problem such as spoken digit classification.

V. The Cocktail Party Problem On EEG Data

In the field of machine learning, the Cocktail Party Problem represents the challenge of segregating and focusing on a singular audio signal from a mixture of sounds, just like a human is generally able to do in a noisy environment. This problem is particularly relevant in the analysis of electroencephalogram (EEG) data, where the goal is to identify and classify neural responses to specific auditory stimuli against the backdrop of complex brain activity.

Specifically, the task at hand is to create a model that is able to classify from multi-channel EEG data if the subject is listening to a speaker on their left or on their right (a binary classification problem). The application of CNN’s and WST deep learning architectures as well offers a promising solution to this problem, leveraging deep learning algorithms to extract and learn from the intricate patterns within EEG data, but it was unfortunately met with setbacks that ultimately ended in this analysis being abandoned.

An attempt was made to make a comparison between a WST and a CNN model solving the cocktail party problem on EEG data, but unfortunately we, the authors of this report,

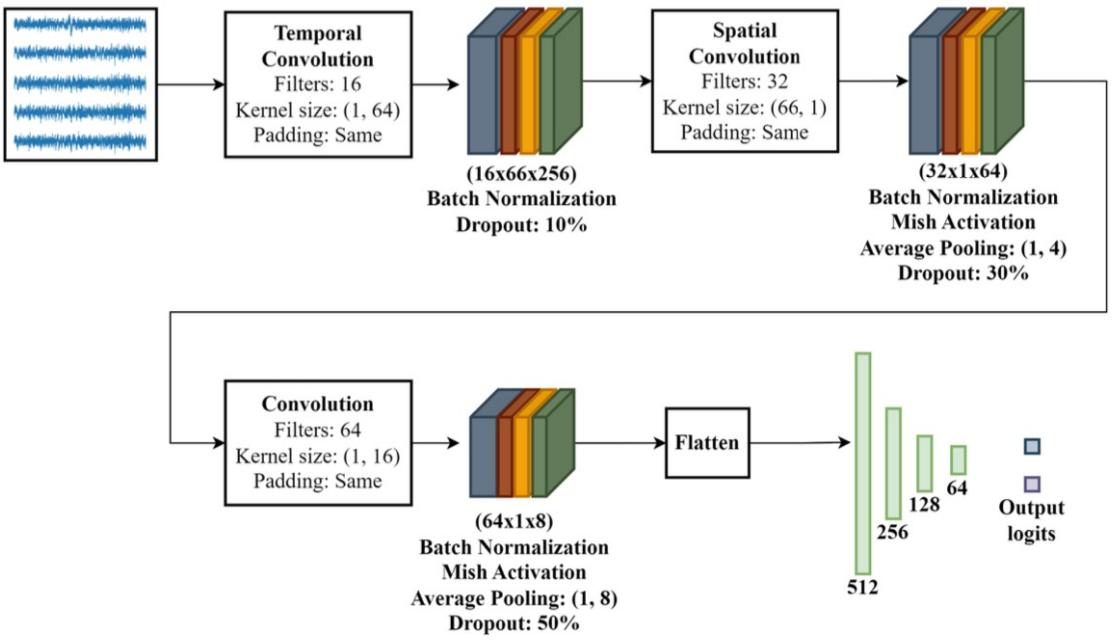

Fig. 18. The CNN model architecture that was attempted to be reconstructed in the cocktail party problem based on A. Tanveer’s paper solving said problem [14]. Note that since the input data is multi-channel both a temporal and spatial convolution is applied.

were not able to recreate a CNN model solving the problem on the data available developed by Tanveer [14]. The CNN model was unable to succeed in the binary classification providing an accuracy varying around 50% corresponding to pure guesses, thus rendering an eventual comparison against a WST model useless.

A. Input data

The data available was provided by Oticon A/S and contains 32 native Danish speakers with hearing loss and are experienced hearing aids users but otherwise healthy and with no other history of neurological disorders. The subjects were equipped with 64 electrodes covering their heads to measure a EEG of them at a sampling frequency of 1024 Hz. Every electrode produces a ”channel”, an unique signal with corresponding time series's. For every subject the measurements were repeated in a total of 80 trials. [14]. Some of these channels are visualized in figure 23.

Interestingly, since there are several channels available the input data could be described as 2 dimensional with 1 temporal dimension and 1 spatial dimension representing the 64 electrodes placed around the subjects head. The information carried in the temporal dimension is thought to be more essential but there is also potential for a deep neural network model to extract information in features carried in the spatial dimension [14].

B. CNN Model Structure

The scope of this project regarding the Cocktail Party problem on EEG data unfortunately ended in the attempt of designing a CNN model that could solve the problem. Several attempts were made to recreate the results of the master thesis paper by M. A. Tanveer replicating the proposed CNN structure solving the problem on the same data in figure 18 [14] but due to an increasing amount of time required to solve the problem and after consultation with our supervisor, Oskar Keding, the decision was made to abandon this application focusing on other applications instead previously covered..

C. Discussion

There are no numerical results or any graphs that can be displayed for the Cocktail Party Problem on EEG data, however there still is room for discussion.

Why was the CNN model not able to extract features from the input data and succeed in application? When training the CNN model it was unable to decrease the error on training data through epochs implying it was not able to extract features depreciating gradient descent (using Adams). Several attempts were made with thorough validation checkpoints along the code pipeline that the problem did not lie in preprocessing steps or in logical errors of handling the multi-channel data before input in CNN model. There is a high certainty that the errors did not lie there and it should be possible to identify patterns in the data because successful classification have been done before. The error is thought to lie in the application of the CNN model somehow. Either the CNN model does not follow the structure in figure 18 as thought or there is something missing not enabling the CNN to extract features properly.

VI. Conclusion

In conclusion, the investigation across two applications demonstrated the effectiveness of the Wavelet Scattering Transform (WST) over Convolutional Neural Networks (CNN) in several key areas. The WST consistently performed similarily or even outperformed CNNs by requiring fewer model parameters and training epochs, particularly in environments with limited training datasets. It also proved capability of robustness against signal corruption and noise, maintaining high accuracy even under challenging conditions, though contrasted to the alternative, the CNN, results vary somewhat. These findings underscore the WST's potential as a more efficient and robust method for automatic feature extraction of low variance features in real-world scenarios, especially where computational resources and data availability are constrained.

VII. References

[1] Mar. 2024. URL: https://en.wikipedia.org/wiki/Convolution.

[2] Jan. 2024. URL: https://en.wikipedia.org/wiki/Rectifier_(neural_networks).

[3] Feb. 2024. URL: https://en.wikipedia.org/wiki/MNIST_database.

[4] URL: https://se.mathworks.com/help/wavelet/ug/wavelet-scattering.html.

[5] Joakim Andén and Stéphane Mallat. "Deep Scattering Spectrum". In: *IEEE Transactions on Signal Processing* 62.16 (2014), pp. 4114–4128. DOI: 10.1109/TSP.2014.2326991.

[6] A. Author. "Multilayer Extreme Learning Convolutional Feature Neural Network Model for the Weak Feature Classification and Status Identification of Planetary Bearing". In: *Journal of Mechanical Engineering* XX.X (2021), pp. XXX–XXX.

[7] Ramesh Reddy Chandrapu et al. "SqueezeVGGNet: A Methodology for designing low complexity VGG Architecture for Resource Constraint Edge Applications". In: *2022 20th IEEE Interregional NEWCAS Conference (NEWCAS)*. 2022, pp. 109–113. DOI: 10.1109/NEWCAS52662.2022.9841955.

[8] Zohar Jackson. *A free audio dataset of spoken digits - an audio version of MNIST.* Aug. 2020. URL: https://github.com/Jakobovski/free-spoken-digit-dataset.

[9] *Kymatio: Fast Scattering Transforms with Deep Learning Frameworks*. Kymatio. 2023. URL: https://www.kymat.io/codereference.html (visited on 04/22/2024).

[10] Sandra Lee, Johnathan Kim, and Samuel Cho. "Deep learning in neuroradiology: A systematic review of current algorithms and approaches for the new wave of imaging technology". In: *Diagnostic and Interventional Imaging* 103.3 (Mar. 2022), pp. 125–133. DOI: 10.1016/j.diii.2021.11.008. URL: https://www.sciencedirect.com/science/article/pii/S1746809421008004.

[11] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. 3rd. Chapter 4, Page 72. San Diego, CA: Academic Press, 2008.

[12] John Muradeli. *Wavelet scattering explanation?* Oct. 2021. URL: https://dsp.stackexchange.com/questions/78512/wavelet-scattering-explanation/78513#78513.

[13] The Test Specimen. *Continuous Wavelet Transform and Convolutional Neural Networks*. Accessed on 2024-04-23. 2023. URL: https://www.thetestspecimen.com/posts/continuous-wavelet-transform-cnn/ (visited on 04/23/2024).

[14] M. Asjid Tanveer. "Deep Convolution Neural Network for Attention Decoding in Multi-Channel EEG with Conditional Variational Autoencoder for Data Augmentation". TFRT-6194, ISSN 0280-5316. MSc Thesis. Lund, Sweden: Lund University, Department of Automatic Control, Mar. 2023.

[15] Wikipedia. *Continuous Wavelet Transform*. Accessed on 2023-04-23. 2023. URL: https://en.wikipedia.org/wiki/Continuous_wavelet_transform (visited on 04/23/2023).

[16] Wikipedia. *Morlet wavelet*. Accessed on 2024-04-23. 2024. URL: https://en.wikipedia.org/wiki/Morlet_wavelet (visited on 04/23/2024).

VIII. Appendix

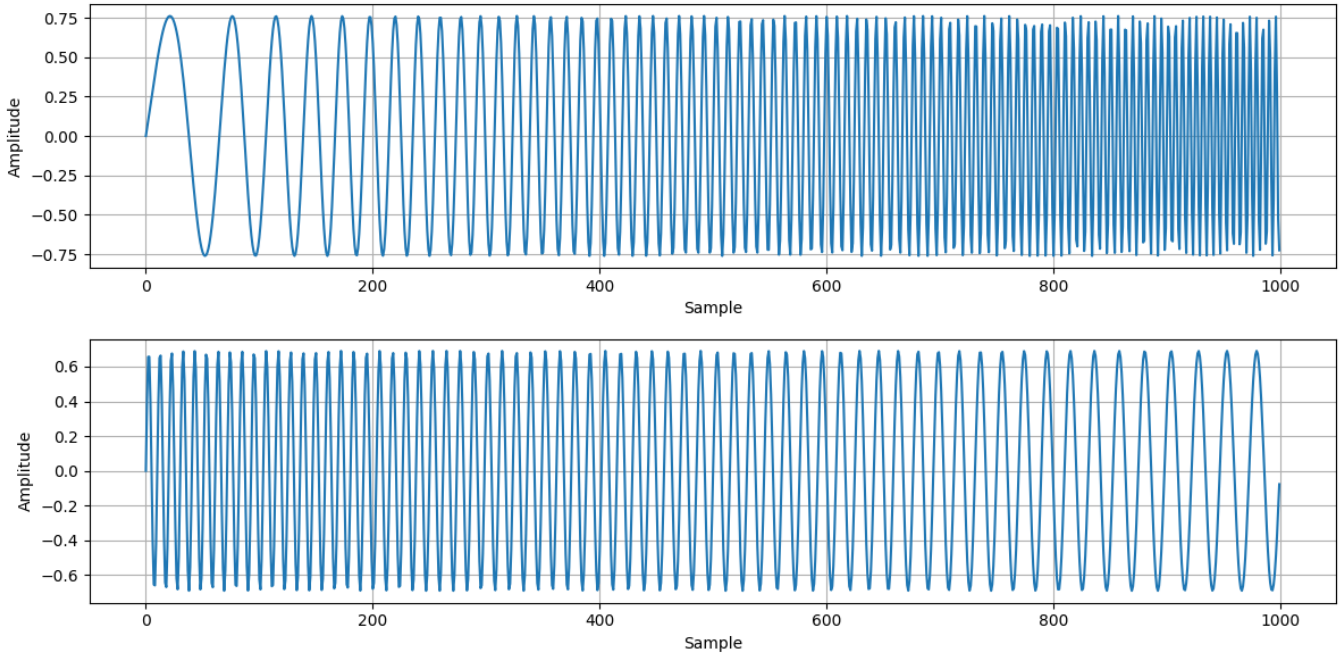

Fig. 19. Illustration of generated unnoisy samples with increasing and decreasing frequency respectively.

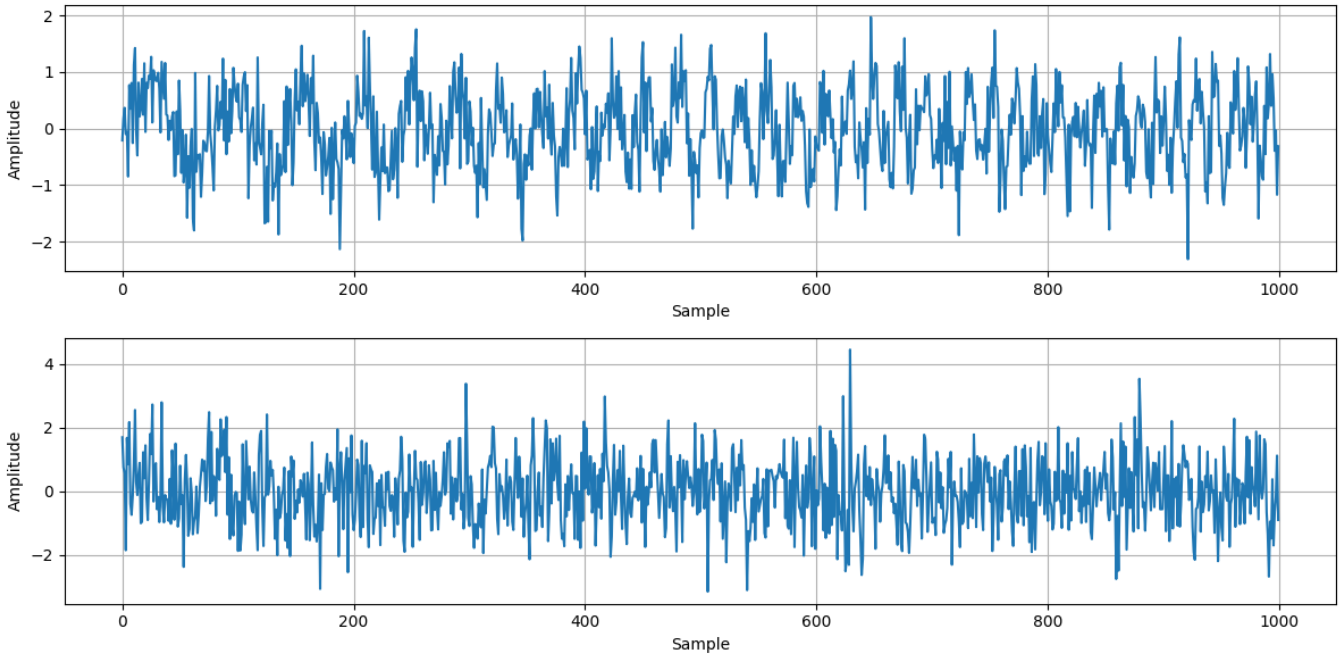

Fig. 20. Illustration of generated noisy samples with increasing frequency. The noise level is 0.5 for the upper figure and 1 for the lower.

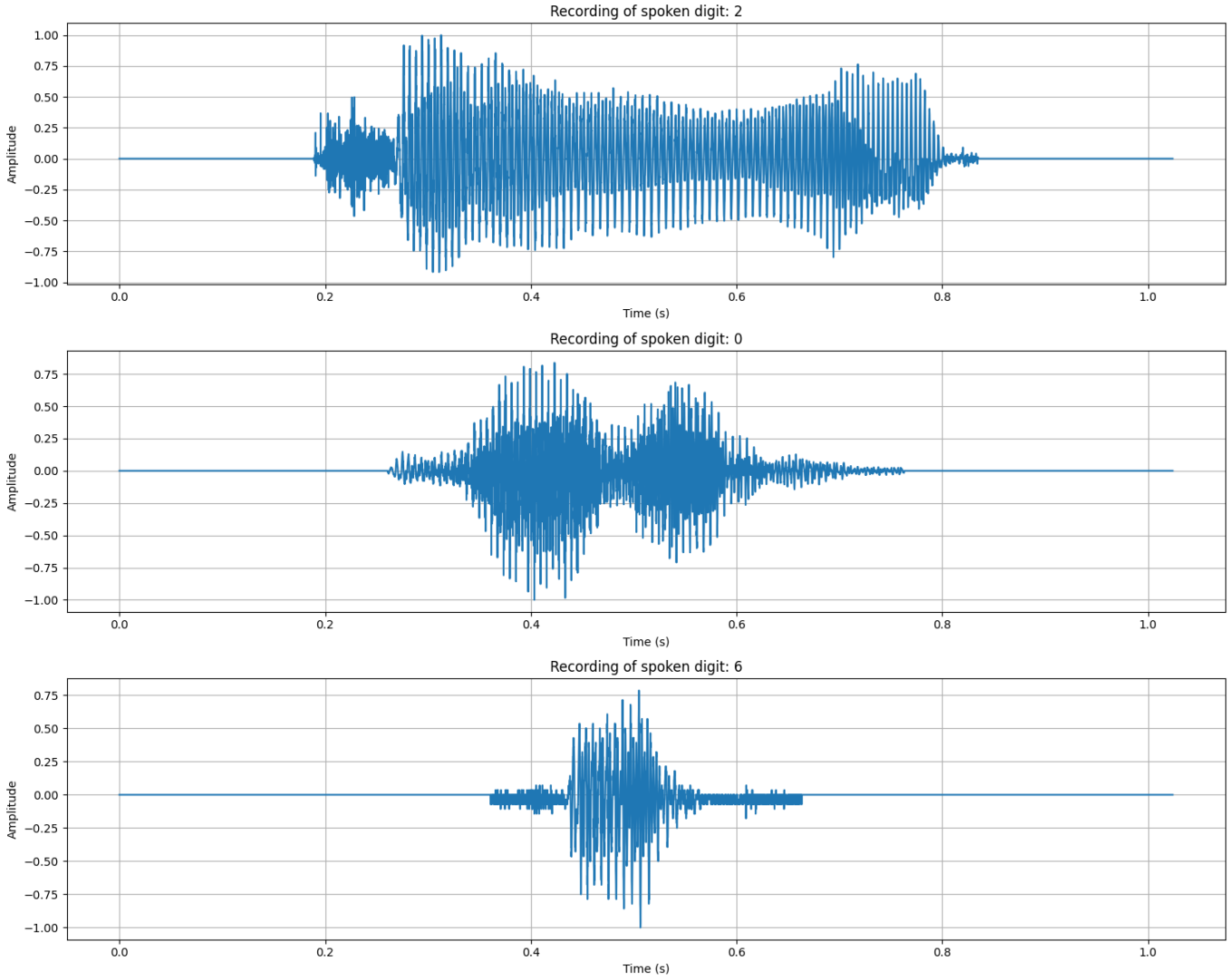

Fig. 21. Representative audio waveforms of spoken digits ’2’, ’0’, and ’6’ after minor preprocessing. The spoken digits are visible from the naked eye, digit 0 showcases this especially with two visible waveforms corresponding to each syllable.

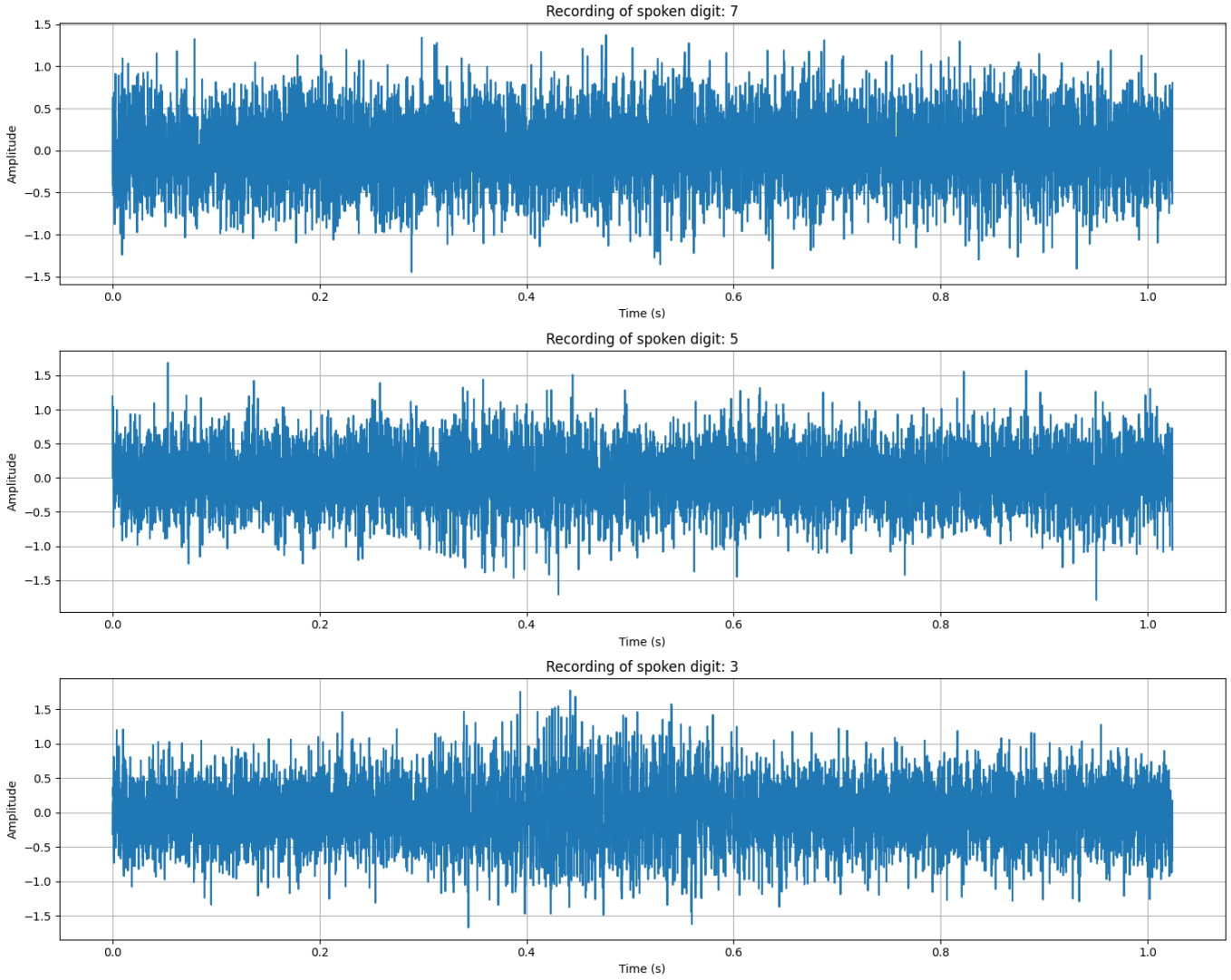

Fig. 22. Representative audio waveforms of spoken digits ’7’, ’5’, and ’3’ with an applied Gaussian noise level of standard deviation $\sigma = 0.4$ and the spoken digit is not visible by the naked eye.

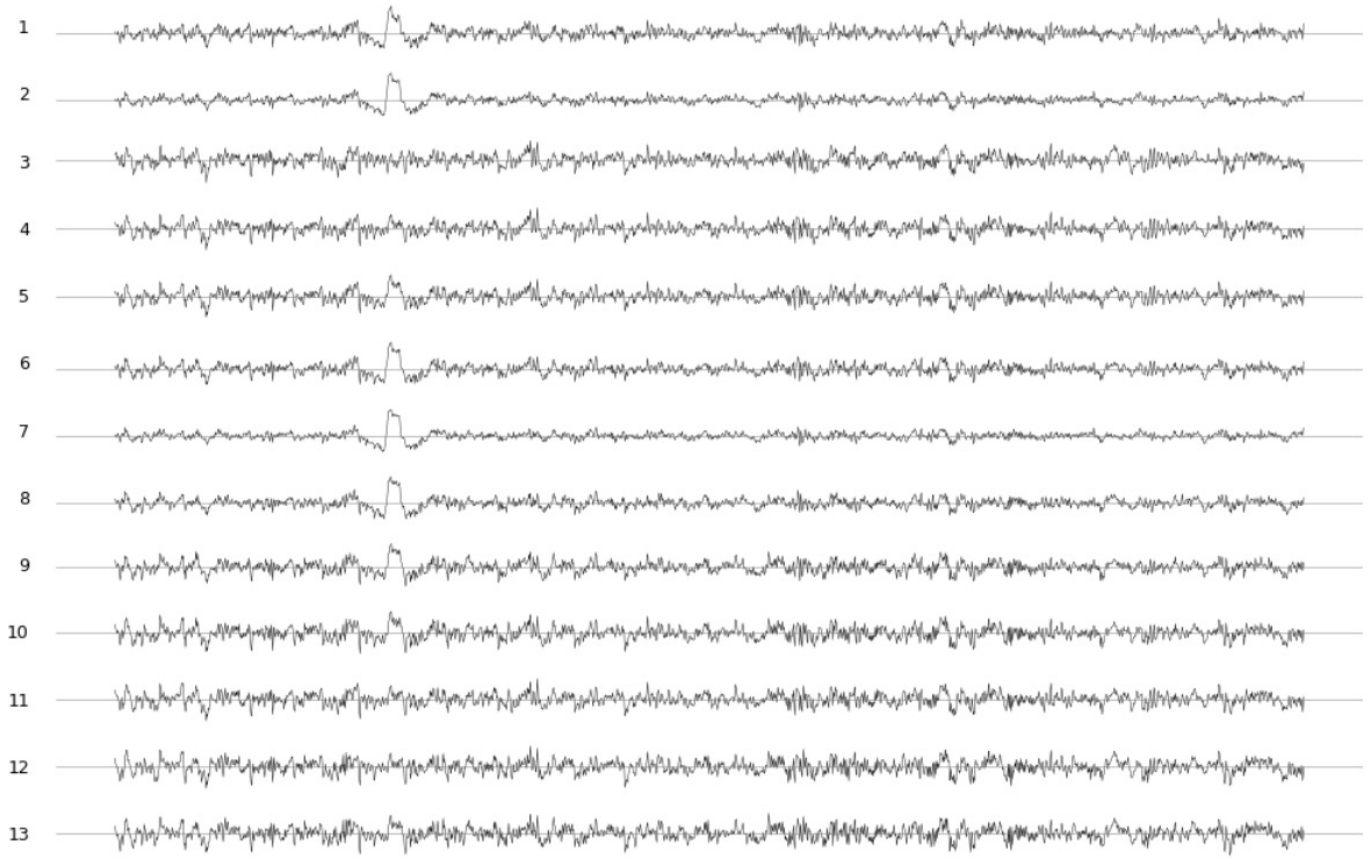

Fig. 23. EEG data channels with time series's from channels 1 to 13. The data is extracted from trial 1 of subject 17. The data has gone trough preprocessing including a bandpass filter, a notch filter, normalization and decimation.