acoustics

Article

Training a Filter-Based Model of the Cochlea in the Context of Pre-Trained Acoustic Models

Louise Coppieters de Gibson [1,2,*] and Philip N. Garner [1]

[1] Idiap Research Institute, 1920 Martigny, Switzerland; pgarner@idiap.ch
[2] EPFL, Swiss Federal Institute of Technology in Lausanne, 1015 Lausanne, Switzerland
* Correspondence: louise.coppieters@idiap.ch

Abstract: Auditory research aims in general to lead to understanding of physiological processes. By contrast, the state of the art in automatic speech processing (notably recognition) is dominated by large pre-trained models that are meant to be used as black-boxes. In this work, we integrate a physiologically plausible (albeit simple filter-based) model of the cochlea into a much larger pre-trained acoustic model for speech recognition. We show that the hybrid system can be trained and evaluated with various combinations of fine-tuning and self-supervision. The results broadly show that the system automatically yields structures that are known to work well. Moreover, these structures lack artifacts that were apparent in (our) previous work using less sophisticated neural models. We conclude that the hybrid structure is an appropriate way to proceed in auditory research, more generally allowing the work to take advantage of larger models and databases from which it would not otherwise benefit.

Keywords: cochlear models; self-supervision; trainable filterbanks

check for updates

Citation: Coppieters de Gibson, L.; Garner, P.N. Training a Filter-Based Model of the Cochlea in the Context of Pre-Trained Acoustic Models. *Acoustics* **2024**, *6*, 470–488. https://doi.org/10.3390/acoustics6020025

Academic Editors: Petr Krysl and Jian Kang

Received: 14 December 2023
Revised: 24 April 2024
Accepted: 13 May 2024
Published: 17 May 2024

1. Introduction

Since the advent of deep learning, the general field of speech technology has advanced to a point that would be unrecognizable to proponents working just a decade ago [1,2]. Perhaps the main difference is that the technology is now driven by the machine learning community rather than the speech processing community. One visible effect of this change is that physiologically inspired approaches that guided the topology of hidden Markov models (HMMs, the previous state of the art) have largely disappeared; they have been replaced by "end-to-end" (E2E) approaches. These in turn learn the representation that leads to the best performance on the available data, irrespective of physiological plausibility. For example, HMMs were typically defined at a phoneme level, but deep architectures have preferred byte-pair encodings [3] or often just (orthographic) letters. Spectral features have been replaced by generic outputs from convolutional layers. The exemplar in these cases is perhaps the "wav2letter" of Collobert et al. [4].

Self-supervision in particular has been a key recent advance in deep learning, in computer vision as well as audio processing [5]. The "self" supervision arises from a loss-function that is designed to reveal discriminability at a given granularity of interest. Crucially, this removes the need for labeled training data, resulting in systems that can be trained on vast amounts of data. To put this into perspective, the LibriSpeech database of Panayotov et al. [6] is one of the largest commonly available labeled speech databases at around 1000 h. By contrast, self supervision is associated with datasets of tens of thousands of hours; the million hours of Parthasarathi and Strom [7] is roughly a century. The commercial side of the community has made pre-trained models available [8,9]. Note that this pre-training implies almost fully trained; it is distinct from the pre-training that used to be applied to some networks to enable them to respond to conventional training [10]. However, subsequent training of the recent models is common and referred to as fine-tuning.

Given that these recent systems work well, the question arises: "what have they learned?". This is difficult to answer because their component parts cannot readily be mapped to biological ones. A recent study has shown that analogies can be drawn between layers of such systems and brain function [11], suggesting that the two fields are in fact converging. In general, however, it seems reasonable that in order to make inferences about biological function, it is necessary to build the deep networks using components that are themselves interpretable. Until quite recently, embedding such components into a deep-learning framework might have been onerous owing to the need for their bein differentiable.

Fortunately, current automatic gradient packages allow networks of arbitrary operations. They also allow arbitrary parts of a network to be trained. This automatic gradient method is a technique that automates the calculation of gradients for the model parameters, enabling stochastic gradient descent computation. It alleviates the need for manual derivation while enhancing the efficiency and scalability of these gradient-based algorithms.

The innovation of this work lies in the transformative impact that a self-supervision model brings to the auditory model. Therefore, we combine a current state-of-the-art pre-trained model with a trainable filter front-end to infer a physiologically plausible function of the human auditory apparatus. More specifically, we build upon a previous study [12] where we showed that attempting to do this with a smaller model led to unexpected results. We show that it is possible to use a state-of-the-art model, and that doing so mitigates the effects of the smaller model.

Section 2 first describes the state of the art in both E2E modeling as well as cochlear modeling for speech technology. Section 3 further details how a simple auditory model can be trained from scratch in the context of a larger model that has been pre-trained. Section 4 contains a logical sequence of experiments showing that:

- Trainable filters can replace the encoder CNN in an already pre-trained model for fine-tuning.
- A physiologically adaptable front-end performs as well as a CNN in a pre-trained model.
- Trainable filters can be incorporated during self-supervision.
- When trained with a large transformer model, SincNet filters do not tend to learn wide-band filters as they do with a smaller MLP model.

2. Background

2.1. Self-Supervised Models

2.1.1. Foundations

Self-supervised learning arose around 2008 in the natural language processing (NLP) field with the model of Collobert and Weston [13]. Self-supervision differs from supervised learning in the way the loss function is computed. Supervised learning necessarily requires labeled data; the output of the network is directly compared to the labels and the difference is back-propagated through gradients. Assigning labels can be onerous. By contrast, self-supervised learning has the advantage of needing no labels to train, meaning it can make use of huge amounts of data. The model is typically trained either as an auto-encoder (the data are the labels) or by contrastive loss (see below).

Self-supervised models rely on a two-stage training procedure: pre-training and fine-tuning. These map onto what might be traditionally called training and adaptation respectively. Pre-training is resource-intensive and the resulting models can be large. A significant recent trend has been for proponents to make such models available online [8]. A core goal of this study is to investigate what can be done in the acoustic research field starting from those available pre-trained networks.

2.1.2. Wav2vec

Wav2vec [5] was the first attempt at applying self-supervision in the sense of BERT to the automatic speech recognition (ASR) field. It was inspired by wav2letter [4] and unsupervised machine translation [14], a model that translates between languages based on two unlabeled datasets. The structure of wav2vec [5] comprises two main blocks: the encoder network and the context network. The encoder consists of a 5-layer convolutional network (CNN) down-sampling the input waveform from 16 kHz to a 100 Hz signal. The context network in wav2vec is also a CNN with kernel sizes of three and strides of one. In vq-wav2vec, Baevski et al. [15] proposed to integrate quantization between the encoder and the context layers. The recent Wav2vec2 [16] parallelizes the quantization and an enhanced version of the wav2vec structure; the encoder CNN is now seven layers deep and down-samples the signal to 50 Hz and the CNN-based context layer is replaced by a transformer. Of those two networks, the transformer stack accounts for 94.4% of the total number of trainable parameters.

Wav2vec models use contrastive loss as an objective during training, and the connectionist temporal classification (CTC) of Graves et al. [17] for decoding. The original contrastive loss of wav2vec was formulated to favor predictability for adjacent observations, or "predictors", with the opposite for more distant ones, hence "distractors". Oord et al. [18] proposed another approach to the contrastive loss computation, this time based on the cross entropy loss. The loss equation in wav2vec2 adopts this more recent approach, being

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \overbrace{\mathbf{q}_t}^{\text{predictor}})/\kappa)}{\sum_{\tilde{\mathbf{q}}\in\mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \underbrace{\tilde{\mathbf{q}}}_{\text{distractor}})/\kappa)} \quad (1)$$

where *sim* is the cosine similarity, $sim(\mathbf{a},\mathbf{b}) = \mathbf{a}^T\mathbf{b}/|\mathbf{a}||\mathbf{b}|$, κ is a constant that regulates the entropy of the cosine similarity, preserving the relative ranks of events. In the machine learning field, the analogous parameter controlling the smoothness of probability distributions is commonly referred to as 'temperature'. The parameter c_t is the context representation and $q \in Q_t$ are vectorized samples of other parts of input waveform in the latent space. The goal of Equation (1) is to find a quantized representation for speech in the latent space, training towards orthogonal representations of the quantizers in that latent space. The model outputs a context representation c_t that is able to guess the true $\mathbf{q}_t$ vector quantization (the predictor) of the latent space out of $K+1$ candidates (with K distractor quantizers and one closely related target; in wav2vec2, 100 distractors are sampled out of the same utterance).

2.2. Cochlear Models

2.2.1. Filterbanks

Cochlear models have been studied for many years, with our current understanding perhaps going back to the work of Von Békésy [19]. Simplistic (but functional) approaches consider the cochlea as a natural filterbank [20,21].

Comparatively recent studies have continued to seek biological plausibility. Lyon [22,23,24], for instance, proposed a model of the complete auditory path where the cochlea is modeled with a cascade of resonators. This model has since been implemented on an FPGA [25] and used in applications such as speaker identification [26] and sound localization [27].

Of particular interest for speech processing is the filterbank scaling. Probably the best known frequency distribution is the mel scale [28], based on frequencies judged to be equally spaced in human perceptual tests. This has been ubiquitous in feature extraction algorithms for ASR, especially in its guise as mel frequency cepstral coefficients (MFCCs). The perceptual linear prediction (PLP) of Hermansky [29] favored the Bark scale, based on noise bandwidths required to mask tones [30–32]. Physical measurements are also possible; the greenwood [33] and ERB (equivalent rectangular bandwidth) [32] scalings lead to more extreme warping than mel.

2.2.2. Current Understanding

Current explanations of the workings of the cochlea are as processes of wave propagation through an active oscillator system [34]. According to our current best understanding, the cochlea works as an array of Hopf oscillators [35,36]. These active oscillators incorporate the interaction between the inner and outer hair-cells with the tectoral and basilar membrane, and explain oto-acoustic emissions [37,38] as arising from the outer hair cells. Several works have implemented those Hopf oscillators as models for the cochlea [39–41].

Notwithstanding, in the present study we limit ourselves to filterbanks, with active systems being a goal for further research work.

2.3. ASR with Trainable Filters

2.3.1. From Cochlear Models to E2E ASR

For many years, ASR front-ends such as MFCC [42] and PLP [29] were loosely based on models of the cochlea. However, given the large number of choices within such models, and parallel success with raw images as input [43], E2E approaches have been investigated for audio processing. Early work involved trainable convolution layers on raw audio inputs [2,44,45]. Although they can perform well, such black box models lack interpretability, explainability, comprehensibility and transparency [46,47]. Retaining an explicit filterbank structure can alleviate these issues. Candidates for the filterbank have included Gamma-tone [48,49], Gabor [50–52], SincNet [53,54], and Spline filters [55]. Zeghidour et al. [50] in particular showed that using trainable filters consistently increased the performance of speech recognition compared to a model where MFCCs were used. Since we do not have a hypothesis that some trainable filter models would perform better, that this work builds on a previous work using SincNet filters in a previous study, and that there are computational advantages of doing so (see Section 2.3.2), we take the SincNet model of Ravanelli and Bengio [53] as representative of the above models for the present study.

2.3.2. SincNet

SincNet is characterized by a convolutional filter as the first layer in a larger convolutional front-end. In the initial SincNet implementation, this convolutional front-end was followed by a five-layer MLP, whereas in this work, the main idea is to combine this encoder with pre-trained transformers. The implementation of the filter layer uses the Fourier transform property: a convolution in the temporal domain corresponds to a multiplication in the frequency domain. Thus, using a sinc filter as kernel in a convolutional layer gives the filtered signal as output. The name arises because the Fourier transform of a rectangular filter in the frequency domain corresponds to $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$ in the temporal domain. SincNet is implemented as a combination of two low-pass filters $F_1 - F_2$ with cut-off frequencies $f_1 > f_2$, giving a band-pass filter of bandwidth $f_1 - f_2$ and a central frequency of $\frac{f_1+f_2}{2}$. The central frequency and bandwidth are the two trainable parameters. In the temporal domain, the filter is then given by:

$$h[n] = \mathrm{sinc}(2\pi f_2 n) - \mathrm{sinc}(2\pi f_1 n) \tag{2}$$

Ravanelli and Bengio [53] showed that SincNet outperformed MFCCs. They also showed that using SincNet filters gave better results than using only CNN layers. A second study by the same authors showed that the proposed architecture converges faster, performs better, and is more interpretable than standard CNNs [54]. They showed that such trainable filters could map onto specific speech-related features like formants, while standard CNNs did not. SincNet was subsequently incorporated into a joint CTC-attention training scheme by Parcollet et al. [56]. The authors showed that their approach outperforms previously investigated E2E systems.

Based on these examples in the literature that show the reliability of SincNet as a front-end for ASR, we use this trainable filterbank in this work.

2.4. Speech Features

We are particularly interested in the interpretability of the learned filterbanks. Pitch and formants are well known important speech features. For human speech, pitch typically lies between 85 Hz and 300 Hz. Formants are resonances of the vocal tract and define vowel sounds. The first formant is generally situated between 200 Hz and 800 Hz; the second formant lies between 500 Hz and 2500 Hz [57].

2.5. Wide-Band Structures

In a previous study [12], we analyzed the behavior of SincNet filters in a simple neural network structure. Although the literature above had shown that E2E trained filters led to improved ASR performance, we were interested in making inference about filter spacing—whether ASR favored mel, Bark, ERB or greenwood scales. The network we used for this study was the one proposed in the original paper [53] within the pytorch-kaldi framework [58].

Unexpectedly, we observed two types of filters appearing: narrow-band filters and wide-band filters; these are depicted in Figure 1. For narrow-band filters, we identified those that more or less followed the conventional expected frequency bands and covered the whole frequency range. These narrow-band filters showed characteristics of a typical ASR filterbank: it was composed of 30 to 40 filters and the scaling the central frequencies tended to learn is the mel scale.

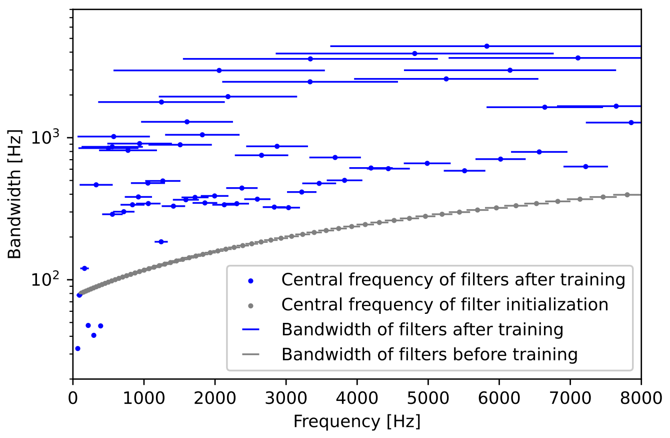


Figure 1. The behavior of filters when training with 60 filters initialized to the mel distribution. The gray bars illustrate the filter initialization with the dots representing the central frequency (on the x-axis) and the bar representing the bandwidth. The blue bars are the filters after training. We observe that two structures are formed: a narrow-band structure, close to the gray filters and covering the whole frequency domain, and a wide-band structure.

By contrast, the wide-band filters were filters learning more unexpected wide-band structures. The reason for this kind of structure was not clear, especially given that, theoretically, wide-band structures could clearly be reconstructed by linear combinations of the other filters. However, literature revealed that some wide-band structures do appear in the human auditory path [59,60].

Additionally, two functions of the filter layer were observed to add artifacts in the frequency domain: the maxpooling and ReLU functions. Maxpooling does not have a physiological equivalent, it is commonly used after convolution operations to reduce the (channel) dimension. ReLU acts as a half-wave rectifier, as do the inner haircels in the cochlea, making the use of this function physiologically relevant. We showed that removing a maxpooling function from the internal structure and keeping the ReLU activation function had a direct impact on the performance of the model, but did not prevent wide-band filters from appearing. One of the goals of this work is to analyze if those wide-band structures still appear when SincNet filters are combined with a large self-supervised network.

3. Method

3.1. Overall Hypothesis

In our previous work (Section 2.5), we showed that a model of the cochlea trained in an E2E manner behaves unexpectedly. The model learns a combination of expected narrow-band structures representative of the cochlea. However, it also learns wider-band structures that are more representative of the higher level auditory pathway. The network used was a combination of a SincNet front-end with a simple MLP as context network and trained in a supervised manner. The state-of-the-art in neural models for acoustic processing of speech is now associated with pre-trained stacks of transformers. Such stacks have been shown to exhibit behavior quite similar to that of the human brain using fMRI [11].

In the following experiments, we aim to show that a plausible model of the cochlea can be combined with an otherwise black-box pre-trained model to yield a model much closer to the human auditory pathway. If such a model continues to behave in the same way as our simplistic one, we could further conclude that something is wrong with our cochlear model; certainly we have no biological evidence for wide-band structure at such a low level. By contrast, if the new model behaves differently, we could conclude that our simplistic model is too simplistic, and that a larger (implying pre-trained) model is *necessary* for such low level auditory investigation.

3.2. Pre-Trained Model

The model used in this work is based on wav2vec2 [16], presented in Section 2.1. For this study, the encoder CNN of wav2vec2 is replaced with SincNet. The encoder layer is composed of SincNet filters followed by a classical CNN of 3 layers as in the SincNet baseline; this CNN down-samples the input signal to 50 Hz, which is then compatible with the further transformer layers. It is illustrated in Figure 2. The training follows the wav2vec2 method, but is customized to take the SincNet filters into account without starting the pre-training from scratch.

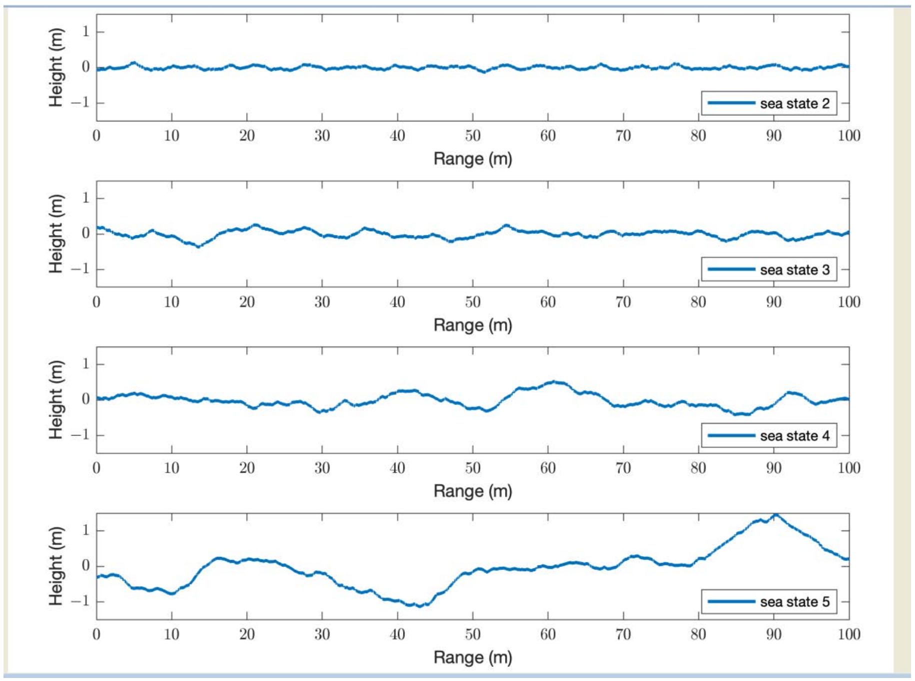


Figure 2. A schematic overview of the original SincNet implementation model used as baseline in our previous work, the wav2vec2 fine-tuning path and the proposed fine-tuning path in this work. Based on compositionality capacity of networks, we combined the feature extractor of the original SincNet model with the pre-trained transformer of wav2vec2.

The framework used is fairseq [8], (which stands for Facebook AI Research in sequence modeling), a framework developed and used by Meta, which has an implementation of wav2vec that can be easily modified. The modifications are described in following sections. Moreover, the code is adapted to run on several GPUs in parallel.

3.3. Dataset

We use the LibriSpeech dataset [6] for all experiments. The amount of data in every subset is given in Table 1 (h: hours, spk: speaker, f: female, m: male, exp: used for given part of experiment, SS: self-supervision, FT: fine-tuning).

Table 1. Summary of LibriSpeech dataset

Subset	h	min./spk	f	m	tot.	exp.
dev-clean	5.4	8	20	20	40	Evaluation
dev-other	5.3	10	16	17	33	Validation
train-100	100.1	25	125	126	251	SS and FT
train-360	263.6	25	439	482	921	SS
train-500	496.7	30	564	602	1166	SS

4. Experiments

4.1. Can Trainable Filters Replace the Encoder CNN in an Already Pre-Trained Model?

Our first hypothesis posits that the CNN encoder block acquires information learnable through trainable filters. To validate this, we replace the front-end with a SincNet encoder (see Figure 2) and compare post-training ASR performance to the baseline. Secondly, by incorporating modifications from our previous work, we expect improved results compared to the original SincNet implementation. Third, the trainable filters in the initial encoder layer should mirror patterns learned by the pre-trained encoder.

4.1.1. Choice of Global Parameters

In order to test this hypothesis, several implementation parameters are adapted: the learning rate scheduler, the structure of the new encoder, the kernel size of the filter layer and the number of updates of the model.

Learning Rate Schedulers

If a pre-trained model is used in, say, ASR, there is normally an adapter layer at the output of the pre-trained model. This layer can be trained whilst keeping the pre-trained model fixed by simply not back-propagating gradients further than the adapter. By contrast, the SincNet that we wish to train is an the input of the pre-trained model; gradients must be back-propagated through the pre-trained model. The new weight value is given by the update parameter learning rule in Equation (3).

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{3}$$

where θ_t is the weight value one time-step before the updated value. $\Delta\theta_t$ is dictated by the equations of optimizer update—in the case of this work, Adam [61].

$$\Delta\theta_t = -\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{4}$$

where:

- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the linear combination of the gradient with first and second moment estimates.
- ϵ is a small constant preventing a division by 0.
- η is the learning rate.

To freeze the model weights, we set the learning rate to zero. According to the gradient descent update Equation (4), if the learning rate η is set to zero, the parameters to which this 0 learning rate is applied are frozen to their initial value, which is a common machine learning practice.

In this work, we adopt a learning rate scheduler integrating a freezing period. During this freezing period, the encoder network is able to learn what the previous CNN layers had learned during the pre-training phase and the context network stays frozen.

The original wav2vec2 implementation uses two types of learning rate schedulers: the polynomial decay and tri-stage learning rate scheduler. The polynomial decay scheduler is used for non-pre-trained network chunks; in the wav2vec framework, this corresponds to the self-supervision phase while the tri-stage learning rate scheduler is used for fine-tuning. For this experiment, the polynomial decay learning rate was assigned to the encoder network (Figure 3c), since this part of the network is replaced by a SincNet encoder and consequently trained from scratch in the experiment. For the context network, we created a hybrid version of the tri-stage learning rate scheduler incorporating a freezing period at the beginning (Figure 3d). The whole training scheme is schematically summarized in Figure 4, clearly depicting the two training phases. Freezing the transformers and projection layer in the first place avoids catastrophic forgetting of the transformer weights. Fairseq's composite optimizer enables distinct learning rate schedulers for different model parts, utilizing a pass-through general optimizer to automatically associate each part with its specific scheduler and optimizer.

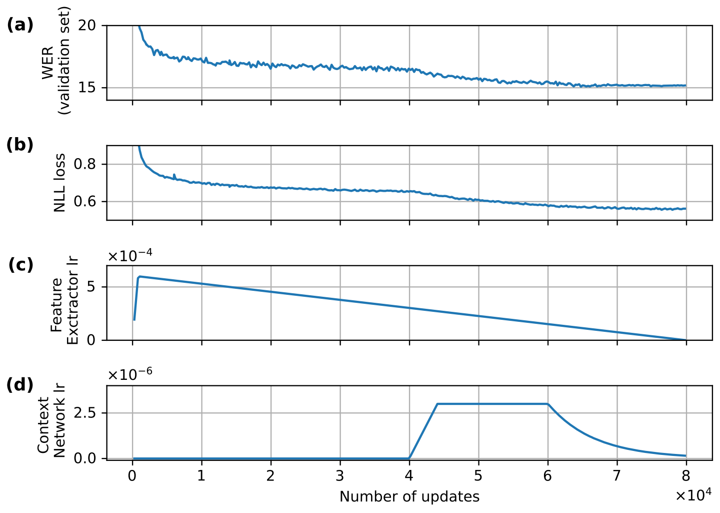


Figure 3. Training curve of the long-run training of 100 k updates. (**a**) depicts the WER of the validation set (dev-other), (**b**) contains the negative log likelihood of the loss, and (**c**,**d**) depict the learning rate evolution of the feature extractor and context network, respectively.

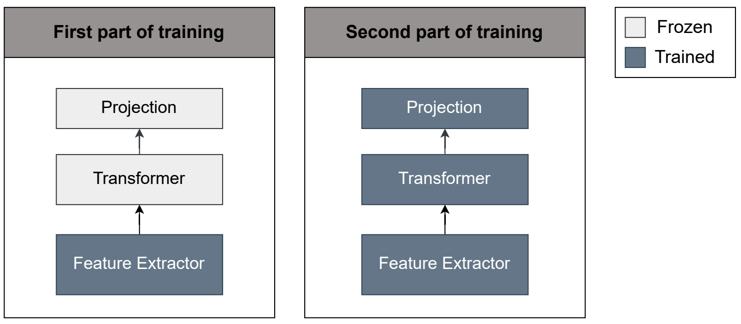


Figure 4. A more schematical explanation of the behavior of the network training explained in Figure 3. There are 2 training phases: one up to 40 k updates where the whole model except the feature extractor is frozen, and the second phase, where everything fine-tunes together.

SincNet Modifications

In the previous work (Section 2.5), we showed that removing the maxpooling layer after the filters was physiologically relevant and improved the performance. In this experiment, we compare the performance of the initial SincNet structure and the adapted structure.

Kernel Size

With the larger model, we observed a saturation effect at lower frequencies in a draft experiment, linked to the maximum filter precision dictated by the kernel size. We replaced the initial kernel size of 129 by 400 in the sinc filter initialization; this both corrected the above issue, and corresponded to the 25 ms used as window size in classical MFCC computations.

Number of Updates

An update corresponds to a back-propagation of gradients in the model, which updates the parameters, while an epoch corresponds to passing the whole dataset through the model. For most of the experiments, the number of updates is fixed to 10 k, which corresponds to 38 epochs with the training set train-clean-100 for the fine-tuning part. For completeness and comparison, we also report one experiment where the model was able to train for 100 k updates, with 40 filters and a kernel size of 400.

4.1.2. Results

The results of this first experiment are summarized in Table 2 and Figure 3. Concerning the performance, the WER results of Table 2 are all below 4%. This means that a SincNet encoder is capable of replacing the original wav2vec2 encoder when keeping the context network fixed.

Table 2. Comparison between a short and long run using no maxpooling after the filter layer in the first 2 experiments and with a maxpooling that has a kernelsize of 3 in the third experiment. Thus, the third experiment has a downsampling of a factor 3 just after the sinc filters via a maxpooling operation in the time domain.

n_filters	Maxpooling	Kernel Size	n_updates	WER
40	-	400	100 k	**3.31**
40	-	400	10 k	3.53
40	3	400	10 k	3.64

The encoding and impact of the learning rate scheduler on the WER and loss curve are illustrated in Figure 3. During the first half of the training updates, the context network is frozen through the zeroing of learning rate, while the encoder network is able to train. In the second half of the experiment, the training of the context network is enabled; the transformer can slightly adapt itself to the trained encoder. In the WER and loss curves, we can see a clear impact of the transformers when they obtain the ability to train around 40 k updates. However, when only enabling the SincNet encoder structure to train, the network already shows a decent performance during the first part of the training.

As summarized in Table 2, three different experiments were performed to address this first hypothesis: two short experiments to analyze the impact of removing the maxpooling layer and a third experiment where the training time was much longer. As in the previous work (Section 2.5), removing the maxpooling function improves the global performance of the ASR. Further, letting the experiment train for a longer period makes the performance increase.

The shape of the filter distribution for the different runs of Table 2 are illustrated in Figure 5. When training the network for 100 k updates instead of 10 k, the WER continues decreasing, but the spatial distribution of the filters does not move much anymore. Changing the network structure by considering the maxpooling layer, however, has a small impact: for lower frequencies, some wider filters are appearing. This means that using maxpooling precludes the wide-band filters to be learned higher up in the network;

nevertheless, the proportion of wide-band filters are much lower than what we obtained in the previous work, and the size of the bandwidth is smaller.

A consistent pattern emerges in the filter distribution shape across various runs within the spectral content, up to approximately 4 kHz. Below 1 kHz, a bump of very small (and consequently precise) filters occurs in the filterbank between 200 Hz and 800 Hz; this frequency range corresponds to the first formant (see Section 2.4). From 1 kHz to 5 kHz, the bandwidths of the filters are all around 400 Hz. Above 5 kHz, the filters seem to learn something different in every run.

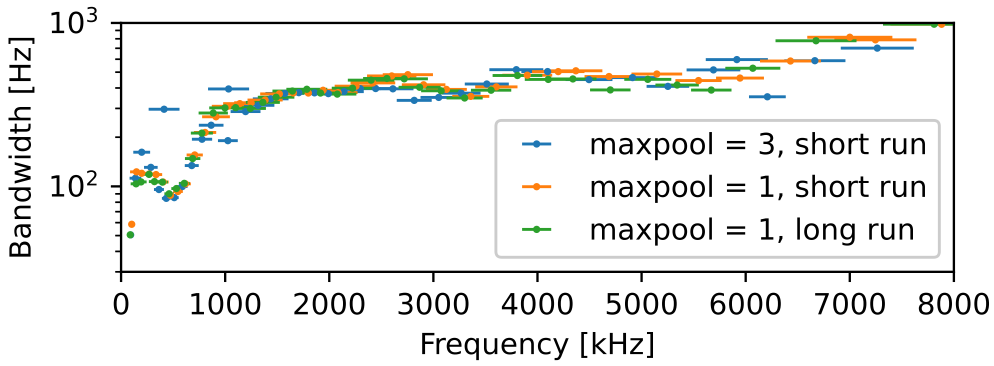


Figure 5. Comparison of filter distribution for different run lengths and internal SincNet structure. Using maxpooling within the filter structure makes some wide-band filters appear in low frequency ranges.

4.2. Does a Physiologically Adapted Front-End Perform as Well as a CNN in a Pre-Trained Model?

4.2.1. Hypothesis

Ravanelli and Bengio [54] showed that using the SincNet model instead of simple CNN converged faster, performed better, and was more interpretable. We hypothesize that if we train a SincNet model and a CNN model with the same number of layers from scratch, combined with a pre-trained context network, the SincNet model would also show those characteristics.

This can be tested by performing and comparing four experiments: The first experiment retrains the wav2vec2 encoder network from scratch to gauge its performance without the pre-training information, setting a baseline for encoder relearning while keeping the pre-training information of the context network. Second, we use the baseline CNN with a layer of the same size as the trainable filters to be comparable with Ravanelli and Bengio [54], but this time combined with a pre-trained context network. The third experiment consists of training a large SincNet encoder. The baseline CNN structure is combined with a layer of trainable filters and trained with the pre-trained transformers. Finally, we compare these experiments with a SincNet structure containing fewer CNN layers (than the structure used in the experiments of Section 4.1).

4.2.2. Results

The convergence speed can be analyzed in the training curve. Figure 6 shows that in terms of training, the SincNet structure converges faster towards an equilibrium compared to a pure CNN structure.Concerning the validation, this is true in the very beginning, especially concerning the CNN with the same shape as SincNet (red curve) up to 2000 updates. We based our analysis of the convergence speed on the update metric to align the loss on the amount of data that go through a forward–backward pass. However, to be complete, Table 3 details the time that every experiment takes and the number of parameters that are contained in the front-end for purposes of comparison.

Table 3. Table summarizing the time of each experiment on 4 parallel GPUs and the number of parameters contained in the feature extractor. Note that this number of parameters is quite small compared to the rest of the network (90 M parameters).

Model	Time	N. Parameters
Small CNN	7 h 14 min	0.575 M
Large CNN	9 h 48 min	4.406 M
Relearn CNN (w2v2 shape)	11 h 57 min	4.206 M
Relearn CNN (SN shape)	14 h 7 min	4.422 M

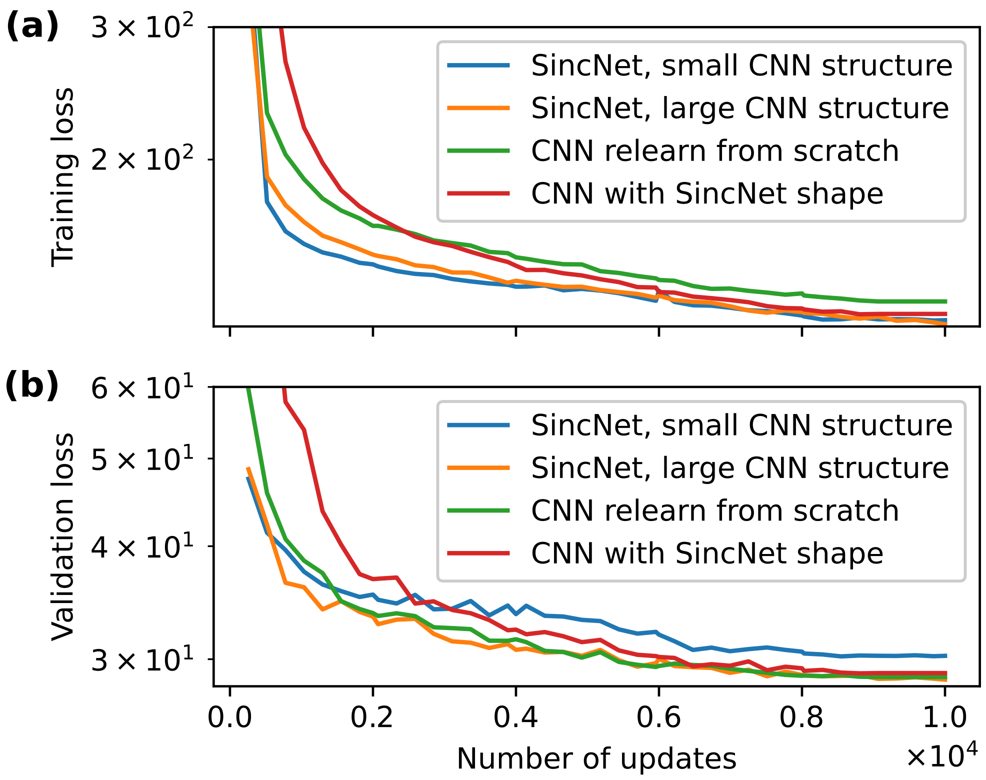


Figure 6. Loss curve during (**a**) training and (**b**) validation.

The performance of the different experiments is summarized in Table 4. Overall, the 95% confidence interval of the large SincNet structure performance overlaps widely with the baseline experiment; this means that the performance with SincNet is not significantly better than the baseline CNN. However, between the SincNet shape CNN and the SincNet implementation, which correspond to the experiments performed in [54], the overlap is less important and SincNet performs slightly better.

Table 4. Performance of the different experiments on the capacity of SincNet (SN) to replace the initial CNN structure with a 95% confidence interval, assuming the result is beta distributed.

	Train Loss	Valid Loss	WER [%]
Relearn CNN (w2v shape)	129.5	28.67	3.35 ± 0.15
Relearn CNN (SN shape)	124.7	28.96	3.45 ± 0.15
SincNet (large CNN)	**120.8**	**28.48**	$\mathbf{3.33 \pm 0.15}$
SincNet (small CNN)	122.3	30.26	3.53 ± 0.15

Finally, concerning the interpretability, Figure 7 illustrates the normalized sum of the filters in the CNN implementation and SincNet after training.

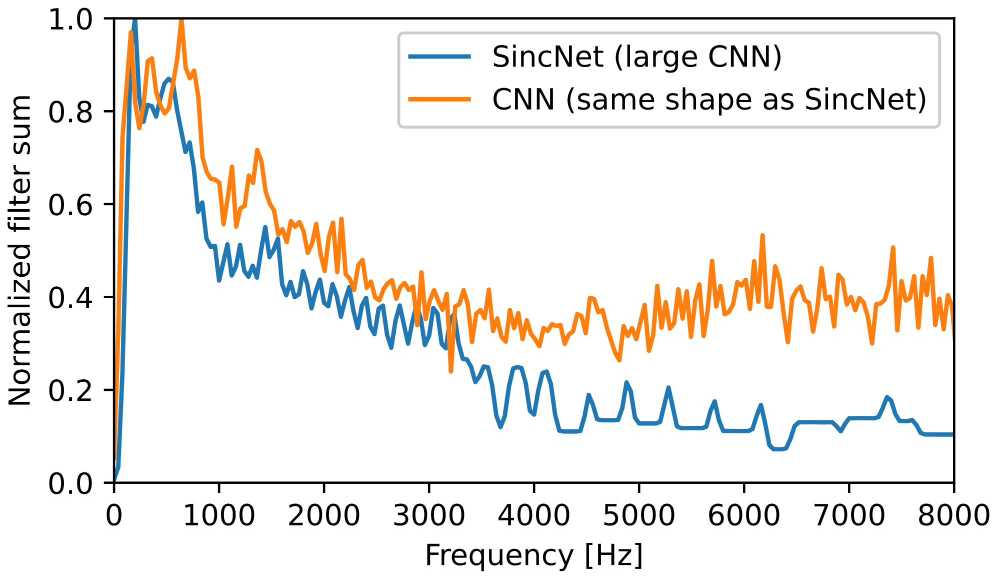


Figure 7. Comparison of the content of the normalized filter sum using SincNet filters and using a same-size generic CNN.

In both cases, "bumps" occur around 700 Hz and 1.4 kHz, which correspond to typical first- and second-formant frequencies. In [54], those interpretable elements in the normalized filter sum were more distinguishable in the SincNet structure than in the CNN structure. Further, for frequencies above 4 kHz, the SincNet filters more efficiently reduce the amount of information recorded in that area than the CNN structure.

To summarize, in the context of a self-supervised model with a pre-trained context network, using a SincNet encoder converges faster and performs as well as the baseline CNN trained from scratch and better than the similar shape CNN. Concerning the interpretability, despite being more interpretable by design, the SincNet structure does not seem to better emphasize interpretable signal components than a CNN using 40 filters.

4.3. Can Trainable Filters Be Incorporated during Self-Supervision?

Another way to train the filters using wav2vec2 is to incorporate the filters into the model in the self-supervision phase.

4.3.1. Hypothesis

Although more computationally demanding, using the self-supervision training path is more faithful to the original wav2vec2 training path and it enables the filters to train on two successive tasks. We hypothesize that continuing the self-supervision task while keeping the context network frozen would lead the filters to learn a similar distribution to the first experiment results. Moreover during self-supervised learning we can either let the filters train or keep them frozen and let them only train during fine-tuning. In this case, we expect the rest of the network to adapt itself to the frozen filter distribution and we do not expect the filters to differ much from what they previously learned during further fine-tuning. However, we expect the free filters to show slightly better performance since they are able to adapt to the fine structure in the frequency range.

Self-Supervised Learning

The first step of this experiment consists of further training the model through the contrastive loss task (see Equation (1)). To be consistant with the original self-supervision dataset we included the two other training subsets of LibriSpeech (see Table 1).

An important parameter to fix is the number of updates needed to obtain a comparable performance as the baseline pre-trained model. Since self-supervision implies a performance measure that is not based on labels, the best comparison measures are the loss and the accuracy.

Table 5 compares the accuracy and loss of the training and validation curves of the baseline and the wav2vec2 model integrating the trainable filters. Using 10 k updates, the accuracy and loss differ from 10–20% from the baseline loss and accuracy. With 100 k updates, the final loss and accuracy match the baseline results up to 1% of error.

Table 5. Comparison of the loss and accuracy of the baseline pre-trained model and the model after a further pre-training for both frozen and trainable filters. After 10 k updates, the model already performs well but not as good as the baseline, while after 100 k, the model reaches the performance of the baseline in terms of loss and accuracy.

	Number of Updates	**Accuracy [%]**		**Loss**	
		Train	**Valid**	**Train**	**Valid**
Baseline	0	61.1	64.6	2.10	1.96
Frozen filters	10 k	52.2	62.2	2.49	2.13
Trained filters	10 k	56.9	63.0	2.34	2.06
Frozen filters	100 k	60.5	66.5	2.13	1.87
Trained filters	100 k	**61.4**	**67.2**	**2.09**	**1.84**

Fine-Tuning

The second step consists of fine-tuning this self-supervised model. In this fine-tuning experiment, the learning rates do not have to be disentangled as we did in the first type of training path, since the whole model has been pre-trained.

4.3.2. Results

The final performance is given in Table 6. For a fine-tuning of 10 k updates, the performance is best when the filters have been through a self-supervision task before fine-tuning (see Table 2).

Table 6. Performance of model using trainable filters in pre-training (100 k updates) and fine-tuning (10 k updates).

Pre-Training Phase	**Fine-Tuning Phase**	**WER [%]**
Frozen filters	Trainable filters	3.40
Trainable filters	Trainable filters	**3.37**

Since during self-supervised training, the filters are able to train, they can be visualized both after pre-training and fine-tuning (Figure 8). For the trainable filters, below 1 kHz, a couple of filters become very narrow-band, then around 1 kHz, 2 kHz and 3 kHz, the filters show a narrow-band bump in their structure as if they sought slightly more fine-grained structures at those specific frequencies. Filters do barely move between during self-supervised training and fine-tuning. This means that the rest of the model is probably more inclined to move towards a better suited equilibrium to minimize the loss. Moreover, globally the shape of the filter distribution shows similar behavior between the two different training paths.

In summary, using only supervised fine-tuning gives a broader flexibility for experiments where several parameters have to be changed, since only a fine-tuning run has to be adapted. Training through both self-supervision pre-training and supervised fine-tuning is more time and resource consuming, but better corresponds to the general idea of using both self-supervision and fine-tuning for training a model.Overall, the filters, when able to train before the transformer adaptation, tend to learn similar patterns.

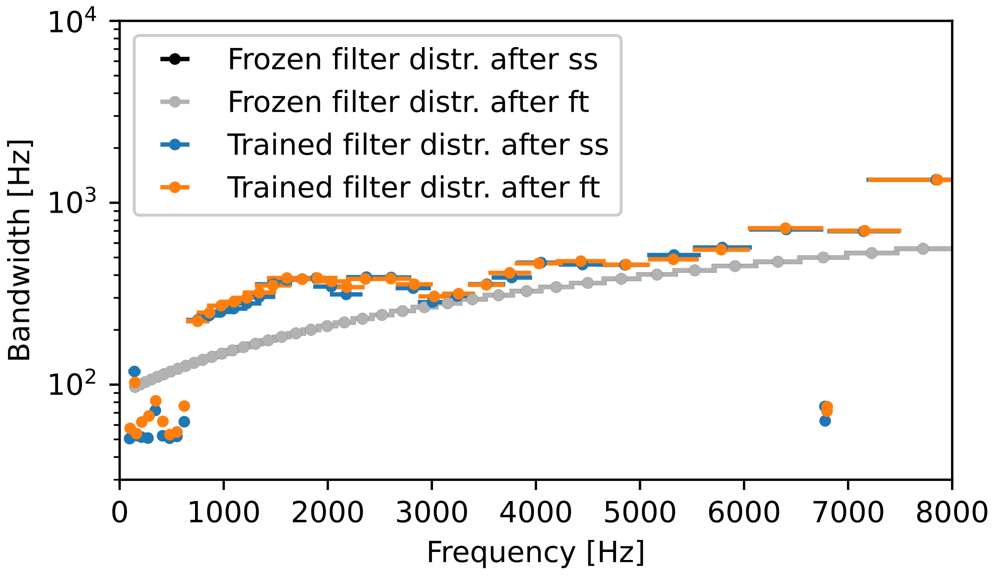


Figure 8. Filter distribution after the self-supervised (ss) training and fine-tuning (ft) of the wav2vec2 model. Globally, filters do not tend to move significantly during fine-tuning when they have been incorporated during pre-training.

4.4. Do Wide-Band Filters Appear in Some Other Training or Model Configurations?

An observation in conflict with our previous work (Section 2.5) is that no wide-band filters appear within the current implementation. Figure 1 showed that some filters learned a very broad-band structure, while in Figure 5, for example, no such wide-band structures appear. Two hypothesis were apparent to explain why those filters could potentially not appear: the number of filters could be too low within the context of a self-supervised model and the freezing of the transformers by decoupling the learning rate scheduler could possibly preclude those filters from appearing.

4.4.1. The Hypothesis of Too Few Filters

Hypothesis

The first hypothesis suggests that a limited number of filters may hinder the emergence of wide-band structures. Additionally, the absence of maxpooling precludes the appearance of wide-band filters. This hypothesis can be tested by increasing the number of filters. If the number of filters is insufficient for the manifestation of wide-band filters, a larger number of them should lead to the emergence of these filters.

Results

Figure 9 illustrates the structure that the filters learn when initialized to 100, 80, 60, and 40 filters. Globally, below 1 kHz, filters learn a very narrow-band frequency-specific filters, above 1 kHz, filters have a bandwidth around 400–500 Hz for all the different numbers of initialized filters. Compared to the previous work, below 1 kHz, the number of very narrow-band filters is much higher in this experiment.

A few wide-band filters do appear when the model is initialized with a high number of filters (80–100), For a smaller number of filters, (40, for example), the wide-band structures as we had in our previous work (Section 2.5) do not appear anymore. Concerning the convenient number of filters to use based on this filter distribution analysis, 40 filters seem to be a convenient number to describe the whole frequency range. This corresponds with the conclusions of our previous paper and it is consistent with choices made in the literature, e.g., [50].

To be complete, we also computed the performance for this model with the different numbers of filters, summarized in Table 7. Similarly to the observations made in our previous work [12], the performance increases slightly with the number of filters we initialize.

Table 7. Performance on the dev-clean subset of LibriSpeech for different numbers of filters.

n_filters	Maxpool	Kernel Size	WER
40	3	400	3.64
60	3	400	3.61
80	3	400	3.57
100	3	400	3.56

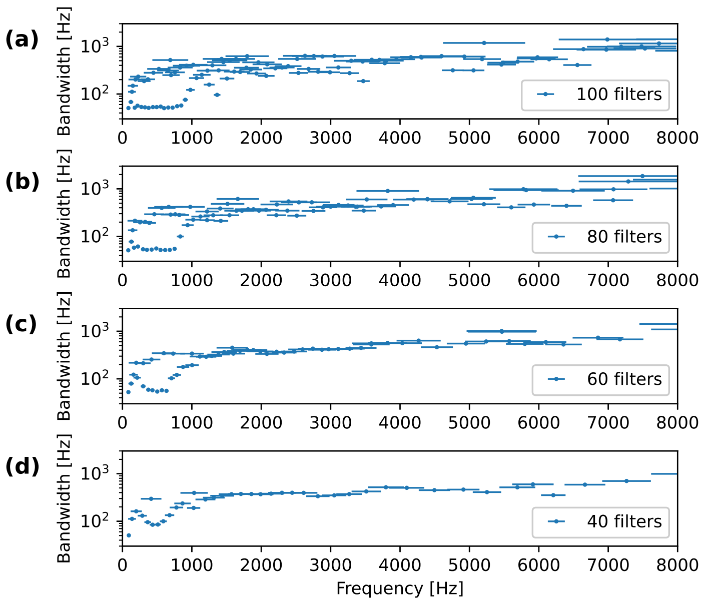


Figure 9. Distribution of the different filters in the frequency domain. The dots represent the central frequencies and the lines represent the bandwidths, similarly to Figure 1. The number of initial filters are (**a**) 100, (**b**) 80, (**c**) 60, and (**d**) 40.

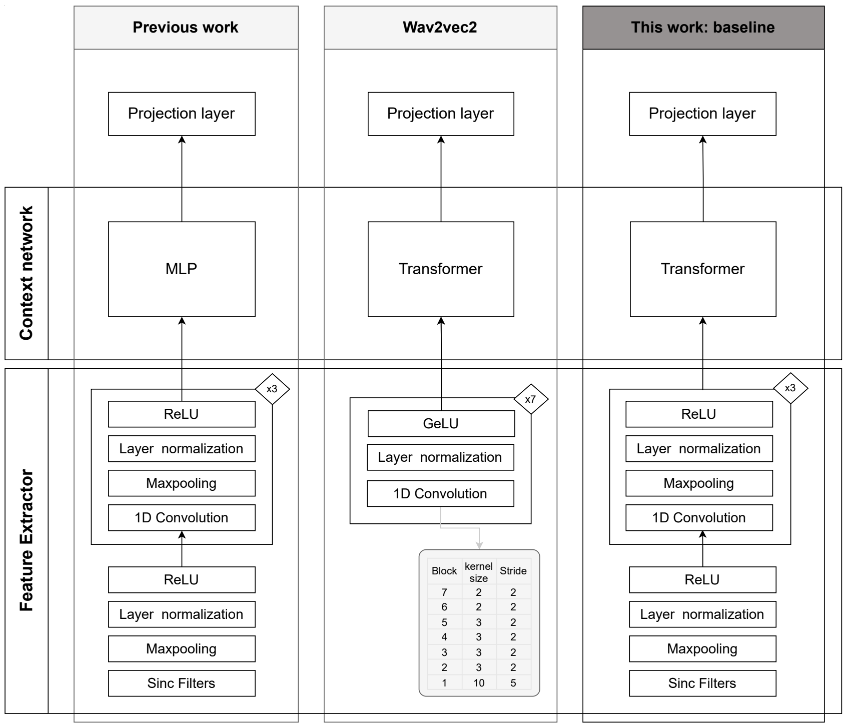
4.4.2. The Transformers Precluding the Filters to Learn Wide-Band Hypothesis

The second hypothesis for why the wide-band filters do not appear is linked to the training path: the transformers are first frozen before being able to train in parallel to the filters, while in the previous work, the whole network trained together. Until now, all experiments have been conducted by first keeping the transformers frozen for a given amount of time in order to train only the encoder network while keeping the transformers fixed. In order to verify this hypothesis, we perform an experiment where all learning rate schedulers start the warm-up period at the same time.

Results

When filters and transformers are free to train jointly from the pre-trained transformer version, the filter distribution does not learn wide-band structures and, moreover, it shows a similar distribution to previous experiments (Figure 10). This means that, used with a pre-trained transformer, the filters do not tend to get a wide-band structure as we had in (Section 2.5).

In summary, when incorporating SincNet into a self-supervised model, the obtained filter distribution does not correspond to the results of our previous work. Only a very small number of wide-band filters appear when enlarging the total number of filters. Besides, the experiments confirmed that the number of filters needed to cover the frequency spectrum in ASR tends to 40. We conclude that a pre-trained context network probably encodes the combination of those wide-band structures, precluding those structures from appearing on the trainable filter layer.



Block	kernel size	Stride
7	2	2
6	2	2
5	3	2
4	3	2
3	3	2
2	3	2
1	10	5

Figure 10. Filter distribution after training.

5. Conclusions

This study proposes an integration of two physiologically grounded concepts: trainable filters and self-supervision. It begins by delineating these concepts in Section 2 and subsequently elaborates on their joint training using a self-supervised context network while concurrently training a new front-end in Section 3. Section 4 outlines a logical sequence of experiments aimed at exploring the behavior of the trainable filters within this framework.

In the first experiment, we demonstrate that a new encoder can be trained while retaining the information acquired during pre-training in the transformer, avoiding a catastrophic forgetting of the transformer weights and resulting in performance close to the state of the art. Additionally, substituting the CNN encoder with a SincNet encoder sheds light on the information expectation of the transformer from the CNN, emphasizing the interest in frequency information at the trainable filter layer.

The second experiment illustrates that, when trained from scratch, the SincNet encoder converges more rapidly than the CNN. However, there is no significant performance improvement, and interpretability across the entire frequency spectrum does not reveal more speech artifacts compared to a CNN.

In the third experiment, we observe that the sole fine-tuning training path offers greater flexibility and is better suited for conducting experiments. While the pre-train–fine-tune training path aligns more with the original concept of self-supervision, it necessitates a larger dataset and substantially more time for the pre-training phase.

Lastly, in the fourth experiment, we investigate why wide-band filters cease to emerge. Through several experiments analyzing behavior with additional filters and employing different training paths, it appears that the pre-trained context network inhibits the emergence of wide-band filters in the initial layer.

Overall, this study demonstrates the feasibility of integrating and fine-tuning state-of-the-art networks with physiologically plausible models. Furthermore, the utilization of decoupled learning rate schedulers enables the fine-tuning of specific parts of the model. We posit that implementing more intricate physiological models and leveraging this approach can facilitate a deeper understanding of how physiological mechanisms may evolve to better interpret external inputs in the encoder network.

Author Contributions: L.C.d.G. performed the experiments, did much of the literature search, and wrote the bulk of the manuscript. P.N.G. supervised the experimental work, and assisted with literature and writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NCCR Evolving Language, a National Center of Competence in Research, funded by the Swiss National Science Foundation grant number 180888.

Data Availability Statement: The data presented in the study is publicly available. The LibriSpeech dataset can be found at http://www.openslr.org/resources/12/ (accessed on 1 November 2023).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Seide, F.; Li, G.; Yu, D. Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In Proceedings of the Interspeech Conference, Florence, Italy, 27–31 August 2011; pp. 437–440.
2. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
3. Sennrich, R.; Haddow, B.; Birch, A. Neural Machine Translation of Rare Words with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2015; Volume 1, pp. 1715–1725. [CrossRef]
4. Collobert, R.; Puhrsch, C.; Synnaeve, G. Wav2letter: An end-to-end convnet-based speech recognition system. *arXiv* **2016**, arXiv:1609.03193.
5. Schneider, S.; Baevski, A.; Collobert, R.; Auli, M. Wav2vec: Unsupervised Pre-training for Speech Recognition. *arXiv* **2019**, arXiv:1904.05862. [CrossRef].
6. Panayotov, V.; Chen, G.; Povey, D.; Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 19–24 April 2015; pp. 5206–5210. [CrossRef]
7. Parthasarathi, S.H.K.; Strom, N. Lessons from Building Acoustic Models with a Million Hours of Speech. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 6670–6674. [CrossRef]
8. Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; Auli, M. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In Proceedings of the NAACL-HLT: Demonstrations, Minneapolis, MN, USA, 2 June 2019. [CrossRef]
9. Babu, A.; Wang, C.; Tjandra, A.; Lakhotia, K.; Xu, Q.; Goyal, N.; Singh, K.; von Platen, P.; Saraf, Y.; Pino, J.; et al. XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale. *arXiv* **2021**, arXiv:2111.09296.
10. Dahl, G.; Yu, D.; Deng, L.; Acero, A. Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition. *IEEE Trans. Audio Speech Lang. Process.* **2012**, *20*, 30–42. [CrossRef]
11. Millet, J.; Caucheteux, C.; Orhan, P.; Boubenec, Y.; Gramfort, A.; Dunbar, E.; Pallier, C.; King, J.R. Toward a realistic model of speech processing in the brain with self-supervised learning. In Proceedings of the Advances in Neural Information Processing Systems 35 (NeurIPS 2022), New Orleans, LA, USA, 10–16 December 2022.
12. Coppieters de Gibson, L.; Garner, P.N. Low-Level Physiological Implications of End-to-End Learning of Speech Recognition. *Proc. Interspeech* **2022**, 749–753. [CrossRef]
13. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning (ICML), Helsinki, Finland, 5–9 July 2008; Association for Computing Machinery: New York, NY, USA, 2008; pp. 160–167. [CrossRef]
14. Lample, G.; Conneau, A.; Denoyer, L.; Ranzato, M. Unsupervised machine translation using monolingual corpora only. *arXiv* **2017**, arXiv:1711.00043.
15. Baevski, A.; Schneider, S.; Auli, M. vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations. *arXiv* **2019**, arXiv:1910.05453.
16. Baevski, A.; Zhou, Y.; Mohamed, A.; Auli, M. Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In *Proceedings of the Advances in Neural Information Processing Systems 33 (NeurIPS 2020), Virtual, 6–12 December* 2020; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 12449–12460.
17. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning (ICML), Pittsburgh, PA, USA, 25–29 June 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 369–376. [CrossRef]
18. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv* **2018**, arXiv:1807.03748.
19. Von Békésy, G. *Experiments in Hearing*; McGraw-Hill: New York, NY, USA, 1960.

20. Geisler, C.D. *Mathematical Models of the Mechanics of the Inner Ear*; Springer: Berlin/Heidelberg, Germany, 1976; pp. 391–415. [CrossRef]
21. Zwislocki, J. Review of recent mathematical theories of cochlear dynamics. *J. Acoust. Soc. Am. (JASA)* **1953**, *25*, 743–751. [CrossRef]
22. Lyon, R.F. Cascades of two-pole–two-zero asymmetric resonators are good models of peripheral auditory function. *J. Acoust. Soc. Am.* **2011**, *130*, 3893–3904. [CrossRef] [PubMed]
23. Lyon, R.F. Using a Cascade of Asymmetric Resonators with Fast-Acting Compression as a Cochlear Model for Machine-Hearing Applications. In *Proceedings of the Autumn Meeting of the Acoustical Society of Japan*; Acoustical Society of Japan: Tokyo, Japan, 2011; pp. 509–512.
24. Lyon, R.F. *Human and Machine Hearing: Extracting Meaning from Sound*; Cambridge University Press: Cambridge, UK, 2017.
25. Thakur, C.S.; Hamilton, T.J.; Tapson, J.; van Schaik, A.; Lyon, R.F. FPGA Implementation of the CAR Model of the Cochlea. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, VIC, Australia, 1–5 June 2014; pp. 1853–1856. [CrossRef]
26. Islam, M.A.; Xu, Y.; Monk, T.; Afshar, S.; van Schaik, A. Noise-robust text-dependent speaker identification using cochlear models. *J. Acoust. Soc. Am. (JASA)* **2022**, *151*, 500–516. [CrossRef] [PubMed]
27. Xu, Y.; Afshar, S.; Wang, R.; Cohen, G.; Singh Thakur, C.; Hamilton, T.J.; van Schaik, A. A biologically inspired sound localisation system using a silicon cochlea pair. *Appl. Sci.* **2021**, *11*, 1519. [CrossRef]
28. Pedersen, P. The mel scale. *J. Music. Theory* **1965**, *9*, 295–308. [CrossRef]
29. Hermansky, H. Perceptual Linear Predictive (PLP) Analysis of Speech. *J. Acoust. Soc. Am.* **1990**, *87*, 1738–1752. [CrossRef] [PubMed]
30. Smith, J.O.; Abel, J.S. Bark and ERB bilinear transforms. *IEEE Trans. Speech Audio Process.* **1999**, *7*, 697–708. [CrossRef]
31. Moore, B.C.; Glasberg, B.R. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. *J. Acoust. Soc. Am.* **1983**, *74*, 750–753. [CrossRef] [PubMed]
32. Zwicker, E. Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen). *J. Acoust. Soc. Am.* **1961**, *33*, 248. [CrossRef]
33. Sridhar, D.; Stakhovskaya, O.; Leake, P.A. A Frequency-Position Function for the Human Cochlear Spiral Ganglion. *Audiol. Neurotol.* **2006**, *11*, 16–20. [CrossRef]
34. De Boer, E. On active and passive cochlear models—Toward a generalized analysis. *J. Acoust. Soc. Am.* **1983**, *73*, 574–576. [CrossRef]
35. Hudspeth, A. Making an effort to listen: Mechanical amplification in the ear. *Neuron* **2008**, *59*, 530–545. [CrossRef] [PubMed]
36. Hudspeth, A.; Jülicher, F.; Martin, P. A critique of the critical cochlea: Hopf—a bifurcation—is better than none. *J. Neurophysiol.* **2010**, *1043*, 1219–1229. [CrossRef]
37. Probst, R.; Lonsbury-Martin, B.L.; Martin, G.K. A review of otoacoustic emissions. *J. Acoust. Soc. Am. (JASA)* **1991**, *89*, 2027–2067. [CrossRef] [PubMed]
38. Kemp, D.T. Otoacoustic emissions, their origin in cochlear function, and use. *Br. Med. Bull.* **2002**, *63*, 223–241. [CrossRef] [PubMed]
39. Hamilton, T.J.; Jin, C.; Tapson, J.; van Schaik, A. A 2-D cochlea with Hopf oscillators. In Proceedings of the IEEE Biomedical Circuits and Systems Conference, Montreal, QC, Canada, 27–30 November 2007; pp. 91–94. [CrossRef]
40. Hamilton, T.J.; Tapson, J.; Jin, C.; Van Schaik, A. Analogue VLSI implementations of two dimensional, nonlinear, active cochlea models. In Proceedings of the Biomedical Circuits and Systems Conference, Baltimore, MD, USA, 20–22 November 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 153–156. [CrossRef]
41. Ammari, H.; Davies, B. Mimicking the active cochlea with a fluid-coupled array of subwavelength Hopf resonators. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2020**, *476*, 20190870. [CrossRef]
42. Davis, S.B.; Mermelstein, P. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *28*, 357–366. [CrossRef]
43. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012), Lake Tahoe, NV, USA, 3–6 December 2012.
44. Palaz, D.; Collobert, R.; Doss, M.M. Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *arXiv* **2013**, arXiv:1304.1018.
45. Palaz, D.; Doss, M.M.; Collobert, R. Convolutional neural networks-based continuous speech recognition using raw speech signal. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, USA, 19–24 April 2015; pp. 4295–4299.
46. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef] [PubMed]
47. Chakraborty, S.; Tomsett, R.; Raghavendra, R.; Harborne, D.; Alzantot, M.; Cerutti, F.; Srivastava, M.; Preece, A.; Julier, S.; Rao, R.M.; et al. Interpretability of deep learning models: A survey of results. In Proceedings of the IEEE Smartworld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, San Francisco, CA, USA, 4–8 August 2017; pp. 1–6.
48. Sainath, T.; Weiss, R.J.; Wilson, K.; Senior, A.W.; Vinyals, O. Learning the speech front-end with raw waveform CLDNNs. In Proceedings of the Interspeech, Dresden, Germany, 6–10 September 2015.

49. López-Espejo, I.; Tan, Z.H.; Jensen, J. Exploring Filterbank Learning for Keyword Spotting. In Proceedings of the 2020 28th European Signal Processing Conference (EUSIPCO), Amsterdam, The Netherlands, 18–21 January 2021; pp. 331–335. [CrossRef]
50. Zeghidour, N.; Usunier, N.; Kokkinos, I.; Schaiz, T.; Synnaeve, G.; Dupoux, E. Learning filterbanks from raw speech for phone recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 5509–5513. [CrossRef]
51. Noé, P.G.; Parcollet, T.; Morchid, M. Cgcnn: Complex gabor convolutional neural network on raw speech. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 7724–7728.
52. Zeghidour, N.; Teboul, O.; Quitry, F.d.C.; Tagliasacchi, M. LEAF: A learnable frontend for audio classification. *arXiv* **2021**, arXiv:2101.08596.
53. Ravanelli, M.; Bengio, Y. Speaker recognition from raw waveform with sincnet. In Proceedings of the IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 18–21 December 2018; pp. 1021–1028. [CrossRef]
54. Ravanelli, M.; Bengio, Y. Interpretable convolutional filters with sincnet. *arXiv* **2018**, arXiv:1811.09725.
55. Balestriero, R.; Cosentino, R.; Glotin, H.; Baraniuk, R. Spline filters for end-to-end deep learning. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 364–373.
56. Parcollet, T.; Morchid, M.; Linares, G. E2E-SINCNET: Toward fully end-to-end speech recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 7714–7718.
57. Olive, J.P.; Greenwood, A.; Coleman, J. *Acoustics of American English Speech: A Dynamic Approach*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1993.
58. Ravanelli, M.; Parcollet, T.; Bengio, Y. The PyTorch-Kaldi Speech Recognition Toolkit. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 6465–6469. [CrossRef]
59. Lomakin, O.; Davis, K.A. On the role of the wideband inhibitor in the dorsal cochlear nucleus: a computational modeling study. *J. Assoc. Res. Otolaryngol.* **2008**, *9*, 506–520. [CrossRef] [PubMed]
60. Biebel, U.W.; Langner, G. Evidence for interactions across frequency channels in the inferior colliculus of awake chinchilla. *Hear. Res.* **2002**, *169*, 151–168. [CrossRef] [PubMed]
61. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.