

Unified Learnable 2D Convolutional Feature Extraction for ASR

Peter Vieting¹, Benedikt Hilmes^{1,2}, Ralf Schlüter^{1,2}, Hermann Ney^{1,2}

¹Machine Learning and Human Language Technology Group, RWTH Aachen University, Germany

²AppTek GmbH, Germany

Email: {vieting, hilmes, schluter, ney}@hltpr.rwth-aachen.de

Abstract

Neural front-ends represent a promising approach to feature extraction for automatic speech recognition (ASR) systems as they enable to learn specifically tailored features for different tasks. Yet, many of the existing techniques remain heavily influenced by classical methods. While this inductive bias may ease the system design, our work aims to develop a more generic front-end for feature extraction. Furthermore, we seek to unify the front-end architecture contrasting with existing approaches that apply a composition of several layer topologies originating from different sources. The experiments systematically show how to reduce the influence of existing techniques to achieve a generic front-end. The resulting 2D convolutional front-end is parameter-efficient and suitable for a scenario with limited computational resources unlike large models pre-trained on unlabeled audio. The results demonstrate that this generic unified approach is not only feasible but also matches the performance of existing supervised learnable feature extractors.

Index Terms: speech recognition, feature extraction, raw waveform modeling.

1 Introduction

Current state-of-the-art automatic speech recognition (ASR) models use neural networks for acoustic modeling [1]. In order for the acoustic model (AM) to process speech, the waveform is transformed into an intermediate representation often called features. This process traditionally consists of handcrafted static operations that take inspiration from psychoacoustic research on the human auditory system [2, 3]. Making the feature extraction a learnable part of the neural AM is desirable from a machine learning point of view, as it allows avoiding information loss by reducing the amount of heuristics and can be trained specifically for the needs of the subsequent neural model. While successful approaches have been proposed [4–8], they are typically still inspired by traditional feature extraction methods.

This can manifest itself in a direct influence such as the initialization of parameters using handcrafted filterbanks [5, 6]. Furthermore, parametric methods such as SincNet [9] rely on well-known and handcrafted signal processing techniques to constrain what the neural network can learn. But even when training filters entirely from scratch, a more indirect influence such as the adoption of intuitions or insights from manual investigations may still be present. One example is a network architecture design that resembles the structure of classical feature extractors, e.g. the Gammatone features in [7]. This is also indicated by the observation that the filterbanks learned in this way resemble their classical counterparts [7, 10, 11]. Furthermore, the usage of logarithmic or root activation functions [5–7] is in line with the non-linear human perception of loudness [12] that is considered in classical features [2, 3, 13] while it is not particularly common in neural networks elsewhere. This raises the question, whether these influences are necessary and whether generic architectures are also feasible.

Many modern Conformer-based ASR models contain a convolutional subsampling block right after the feature extraction. This is often implemented using a stack of VGG-style [14] convolutions [15–17]. Previous works on learnable front-ends replaced the classical feature extraction methods 1:1 in the architecture [11]. As a consequence, the typically 1D convolutional feature extractors are followed by the subsequent VGG-style 2D convolutional module. From the perspective of architecture design, this

results in a discrepancy between the different layer types, as the feature extractors and the VGG blocks do not blend in naturally.

There are two ways of unifying the currently used 1D- and 2D convolutional architectures. The first is to remove the 2D layers and extend the generic 1D layers to also perform the subsampling to the final frame rate on which the AM operates. This results in an architecture like the one used as the feature extractor in the wav2vec model family [18, 19]. Section 2.1 describes it in more detail. Using the wav2vec feature extractor as a learnable front-end for standard supervised ASR has been studied in [11, 20]. These works refer to the usage of the convolutional feature extractor of wav2vec as a replacement for classical feature extraction, to keep the feature extraction lightweight compared to the AM. The Transformer part, which makes up for most of the parameters of the wav2vec 2.0 model, is not considered. This aligns with our goal to address a scenario with limited resources, where large pre-trained models such as the full wav2vec 2.0 or HuBERT [21] models used for representation learning are prohibitive. Also, no additional unsupervised pre-training is used in this work.

We propose a second option for unification, namely the extension of the 2D convolutional layers to cover the full front-end. In general, VGG-style models were successful by replacing convolutional layers with multiple layers with smaller kernels. This is why we investigate whether this is feasible or even beneficial for learnable feature extraction in ASR. Section 2.2 outlines the proposed concept.

Another aspect of generalization concerns SpecAugment [22] that has become a de facto standard for regularization during training of ASR models. As a further consequence of a 1:1 replacement of traditional features with learnable counterparts, SpecAugment is applied between the feature extractor and the VGG blocks in the remaining neural network [11]. From a general point of view, this is a rather arbitrary position inside the overall model. Furthermore, it has several disadvantages for the learnable features [23]. To avoid the usage of SpecAugment at this deliberate position inside the front-end, we can apply it in the short time Fourier transform (STFT)-domain before the feature extraction as proposed in [23]. More details are provided in Section 3.3.

In summary, this work aims to reduce the influence of traditional feature extraction models on the design of the learnable front-end as much as possible. Beyond final performance, we aim to unify and generalize the architecture of the neural front-end. While similar work in [11] trained learnable features without any access to spectral information for the first time, we take this yet a step further such that not even the structure is based on insights of previous manual investigations. Not only does this mean that our studies combine both the feature extraction and the following VGG-style convolutions into a single unified front-end, but also SpecAugment is applied generically before the feature extraction instead of a rather arbitrary intermediate position.

The contributions of this work are as follows:

- We propose a unified generic feature extraction architecture for ASR based on 2D convolutions that minimizes the influence of handcrafted methods,
- the results show that the proposed architecture is feasible and performs on par with existing learnable feature extractors and
- the analyses reveal that despite the generic design, the front-end shows behaviors that are consistent with long-established observations in the field.

2 Feature Extraction Methods

This section presents the feature extraction methods used in this work. As a classical, non-trainable baseline, we use **log Mel filterbank features**, which are arguably the most commonly used features for ASR nowadays. They are computed by applying the STFT to the waveform, in this work with a window size of 25 ms and a window shift of 10 ms. Subsequently, the squared magnitude is mapped into an 80-dimensional representation using the Mel filterbank. The final features are generated by applying a logarithmic activation.

2.1 Existing Learnable Feature Extractors

All feature extractors are expected to be suitable for a low resource scenario. Using large pre-trained models such as the full wav2vec 2.0 or HuBERT models is thus prohibitive in this context. All learnable front-ends are jointly optimized with the entire AM using the supervised ASR training criterion on labeled data. Here, we outline two existing approaches.

Supervised Convolutional Features: The baseline learnable feature extraction in this work are the supervised convolutional features (SCF) as used in [7, 11, 20, 23]. The features are inspired by Gammatone features as the two convolutional layers resemble the Gammatone filterbank and the Hanning window used in [3]. These layers are randomly initialized and serve as time-frequency-decomposition and temporal integration, respectively. Unlike for Gammatone features, a multi-resolutional temporal integration is facilitated by the usage of multiple filters in the second layer.

The first layer operates on the waveform and consists of 150 filters with a size of 16 ms and a stride of 0.625 ms. The activation function computes the absolute value. The second layer consists of 5 filters with a size of 40 and a stride of 16, producing feature frames with a 10 ms shift. The final feature dimension is 750, as every filter is applied to the 150 output channels of the first layer. The activation function is the 2.5th root which is derived from the 10th root used in Gammatone features and tuned for the learnable features [7]. Layer normalization [24] is applied at the end.

wav2vec Feature Extractor: The wav2vec framework [18, 19] deploys an architecture with a convolutional feature extractor on the waveform. While the general idea is the same in [18] and [19], the configuration differs slightly. We follow the configuration from wav2vec 2.0 [19], which consists of seven 1D convolutional layers with 512 channels. Additionally, group normalization is applied after the first layer and each layer is followed by a Gaussian error linear unit (GELU) activation function [25]. Since the Conformer operates on a frame shift of 40 ms in our work, we add an eighth layer with the same configuration as the seventh layer to add another subsampling factor of two. This results in layers with kernel sizes {10, 3, 3, 3, 3, 2, 2, 2} and strides {5, 2, 2, 2, 2, 2, 2, 2}. Note that when referring to the wav2vec feature extractor, we mean the structure of the first convolutional layers as described above. No other parts such as the quantization or Transformer modules are included here. Also, no pre-training is conducted in this work and everything is trained from scratch with the supervised ASR criterion.

2.2 Generic Supervised 2D Convolutional Features

Our proposed 2D convolutional feature extraction is a generic front-end inspired by the commonly used VGG-style subsampling blocks in neural AMs. Following their structural composition e.g. in [26], it consists of a stack of layers that perform 2D convolution over the time and feature dimensions. The additionally introduced channel dimension is merged into the feature dimension after the last 2D layer. Similar to the VGG architecture or [26], the kernel size is 3×3 . However, unlike [26], no preceding classical feature extraction is used and we apply more convolutional blocks instead. Exactly as many layers with strides of two in the time dimension are used so that the resulting output frame rate is 40 ms. More layers with a stride of one may be added in between. For activation, we use the rectified linear unit (ReLU) function.

Special attention has to be devoted to the first layer. Since

the waveform is single dimensional, the first layer has to generate a feature dimension so that the subsequent layers can perform 2D convolution over time and feature dimensions. This can be done via applying the STFT, using either the magnitude or real and imaginary parts. In the latter case, the two parts are passed through a single 2D convolutional layer each and then summed before passing through the remaining 2D layers.

A second possibility is having a filterbank as a first layer. It can be initialized similar to conventional feature extraction, e.g. with a Gammatone filterbank, and frozen during training. However, to keep the pipeline as generic as possible, we also experiment with a learnable filterbank and random initialization. To optimize the performance, we tune the kernel size and stride used by this filterbank in our experiments.

3 Experimental Setup

3.1 Data

We conduct our experiments on the English dataset LibriSpeech [27]. It consists of 960 h of audiobook recordings for training. As labels, we use phonemes with the phoneme set consisting of ARPABET phoneme symbols without stress marker. In order to generate phoneme sequences for words that are not contained in the given lexicon, we use Sequitur [28]. During recognition, we use the official 4-gram language model (LM) trained on the monolingual corpus data. The final performance is evaluated on the standard dev and test sets provided with the dataset.

3.2 Training

A connectionist temporal classification (CTC) model is used to perform ASR in this work. For models with log Mel and SCF features, the feature extraction is followed by a VGG-style convolutional block, subsampling the frames by a factor of 4. A linear transformation to map the front-end output to the model's dimension is applied for all features. Yet, depending on the feature extraction variant, the input dimension and therefore the size of the linear layer differs drastically. The remaining AM consists of 12 Conformer blocks [1] with relative positional encodings [29]. We use a hidden dimension of 512 and a feed-forward dimension of 2048, resulting in ~ 77 M parameters in the log Mel baseline. For learning rate scheduling, we use a one cycle learning rate starting from $7 \cdot 10^{-6}$ and peaking at $7 \cdot 10^{-4}$ at the middle of training. AdamW [30] with a weight decay of 0.01 is deployed as the optimizer and the gradients' norm is clipped with a value of 1.0. We train the model for 100 full epochs using speed perturbation with factors randomly sampled from {0.9, 1.0, 1.1}. For recognition, we use the last checkpoint and decode our model with Flashlight [31]. Training is feasible on a single consumer GPU with 24 GB VRAM (e.g. Nvidia RTX 3090), meaning the entry barrier for reproduction is low. Our work and code is publicly accessible.¹

3.3 SpecAugment

A prevalent method used for regularization is SpecAugment [22], where randomly selected regions of a given input are masked in the time and feature dimensions during training. In this work, we aim to unify the feature extraction front-end and remove influences from traditional methods. In this context, it is desirable not to apply SpecAugment on an arbitrary intermediate layer. Replacing the classical feature extraction 1:1 with a learnable front-end results in SpecAugment being applied between the feature extractor and the VGG-style convolutional block [11]. In contrast, generic positions are upfront before the feature extraction as in [23] or possibly between convolutional front-end and Conformer similar to the masking in wav2vec 2.0 [19]. In wav2vec 2.0, the masking position is inherently predetermined by the structure of the loss function in self-supervised pre-training and not altered for the supervised fine-tuning.

¹<https://github.com/rwth-i6/returnn-experiments/tree/master/2025-2d-conv-features>

Feature extraction architecture	#Params before Conformer	Spec- Augment	WER [%]			
			dev		test	
			clean	other	clean	other
log Mel	1.4M	Features	2.4	5.2	2.7	5.6
SCF	12.4M		2.6	5.7	2.9	6.0
wav2vec	5.0M	STFT	2.6	5.6	3.0	6.0
2D	2.3M		2.6	5.9	2.9	6.3
	0.3M		2.5	5.5	2.9	5.9
			2.5	5.9	2.9	6.2

Table 1: Overview of different feature extraction methods. For the 2D features, the first layer is a randomly initialized filterbank with a kernel size of 256 and a stride of 10. The number of channels is 128 in the better-performing case and 8 in the parameter-efficient case. It is followed by 6 2D layers with a subsampling factor of 2 each. SpecAugment has been tuned for feature-level and STFT-domain separately, but is consistent across the different extractors.

We deploy the first variant here because it has a few key advantages. As explained in [23], it avoids issues arising from the random order of filters in learnable features. Furthermore, it is not possible for the model to spread information over multiple channels in order to bypass the masking. Finally, the hyperparameters are independent of the feature dimension and do not need to be tuned when changing the feature extractor.

4 Results

First, we present word error rates (WERs) for a comparison of the different described feature extraction methods in Table 1. Unlike previous works [11, 23], there is a clear performance degradation when moving from log Mel features to SCF. In contrast to [23], we do not tune the audio perturbation and only applied speed perturbation with rather limited perturbation factors which might contribute to this observation. Moving SpecAugment to the STFT domain results in the same performance. Surprisingly, the wav2vec feature extractor is performing about 5% relatively worse. This is in contrast with [11, 20] where it slightly outperformed SCF. However, our generic unified 2D convolutional feature extractor performs on par with SCF. This demonstrates that it is possible to largely reduce the influence of handcrafted features and rebuild a generic front-end from scratch.

Moreover, our proposed feature extractor is highly parameter-efficient as demonstrated in Table 1. The reported number of parameters before the Conformer includes the feature extractors, a VGG-style module if applicable (for log Mel and SCF), and the linear layer mapping the output to the hidden dimension of the AM. The number of parameters is significantly lower than for the other learnable front-ends even for the best-performing configuration. With a parameter-efficient configuration, it can be reduced to only 0.3M parameters which is considerably smaller than even for log Mel. Note that the parameters are distributed differently for the SCF and wav2vec front-ends, though. For SCF, the bulk of these parameters is in the linear layer before the Conformer because the feature dimension is so large. However, different methods to reduce the feature dimension clearly degraded the performance in preliminary experiments not presented here. For the wav2vec feature extractor, most parameters are in the convolutional layers.

In addition, the learnable front-ends in Table 1 increase the training time compared to the log Mel baseline. The increases are 106% for SCF, 25% for the wav2vec feature extractor, and 57% or 6% for the proposed 2D convolutional front-end configurations. The number of channels in the first layer thus allows trading off the WER against the training time. Further speedups are possible with higher subsampling in the first layer as studied below.

In the following, a number of ablation studies is presented in order to explore different aspects of the generic feature extraction. First, Table 2 compares different choices for the first layer, which needs to extend the 1D waveform with a feature dimension in addition to the time dimension. For this, we compare the STFT against different filterbank variations. When using the STFT, the real and imaginary parts have the theoretical advantage of avoiding

First layer			WER [%]			
Type	Train-able	Init	dev		test	
			clean	other	clean	other
STFT mag.	No	-	2.5	5.5	2.9	5.9
STFT $[\Re, \Im]$			2.4	5.6	2.9	6.0
Filterbank	Yes	GT	2.5	5.5	3.0	5.9
		random	2.4	5.4	2.9	5.9
			2.5	5.6	2.9	5.9

Table 2: Comparison of different first layer types. All experiments use a window shift/stride that results in a subsampling factor of 10. The STFT uses a window size of 400 samples, the filterbanks use 80 filters of size 256. STFT mag. and $[\Re, \Im]$ denote the magnitude or real and imaginary parts, respectively, GT denotes the Gammatone filterbank. After the first layer, 6 2D layers with stride 2 are deployed.

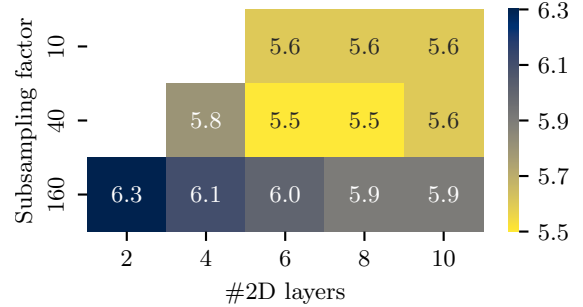


Figure 1: Interaction between subsampling factors in the first layer and number of subsequent 2D layers. All experiments use a randomly initialized learnable filterbank with 80 filters of length 256 here. There are 2/4/6 2D layers with a stride of 2 to obtain an overall subsampling factor of 640 in each case. Further 2D layers are added with a stride of 1. Results are WERs on dev-other.

the information loss that is present in the case of the magnitude. However, this does not translate to better WERs in the experiments. An improved fusion method of both parts could possibly improve the results, but the potential gain is likely to be small.

Moving to a filterbank as first layer, we can choose to train or freeze it during training and to initialize it with a Gammatone filterbank or randomly. As shown in the last three rows of Table 2, the Gammatone initialization slightly helps on the dev sets. However, the random initialization performs on par on the test sets, which is why we conclude that a filterbank of sufficient quality can be learned from scratch even in a generic setup. To align with the goal of a monolithic neural network performing all steps needed for ASR starting from the waveform, we use a randomly initialized filterbank in the following experiments.

Next, we study the interaction between the amount of subsampling performed in the first layer and the numbers of subsequent 2D layers. Figure 1 shows the WER for different combinations of subsampling factors and number of 2D layers. In each row, the leftmost result uses the minimal number of layers with stride 2 required to achieve a final frame shift of 40 ms. Additional layers with a stride of 1 can be added in between to increase the total number of 2D layers. The results show that lower subsampling factors and more layers are beneficial. A plateau is reached after 6 2D layers and subsampling factors 10 and 40 perform similarly there. Unlike for the learnable filterbank, the configuration did not have such a clear impact when using the STFT magnitude in experiments not presented here.

Our last study deals with the kernel size and the number of channels in the filterbank, investigating how much we can reduce the number of parameters without significantly degrading the performance. The results are shown in Figure 3. Reducing the kernel size from 400 –which is the STFT window size in our experiments– to 256 –which is the kernel size in the first SCF layer– and further below does not have a significant impact on the performance. Only dropping to a size of 16 introduces a severe impairment. Reducing the number of channels degrades the WER

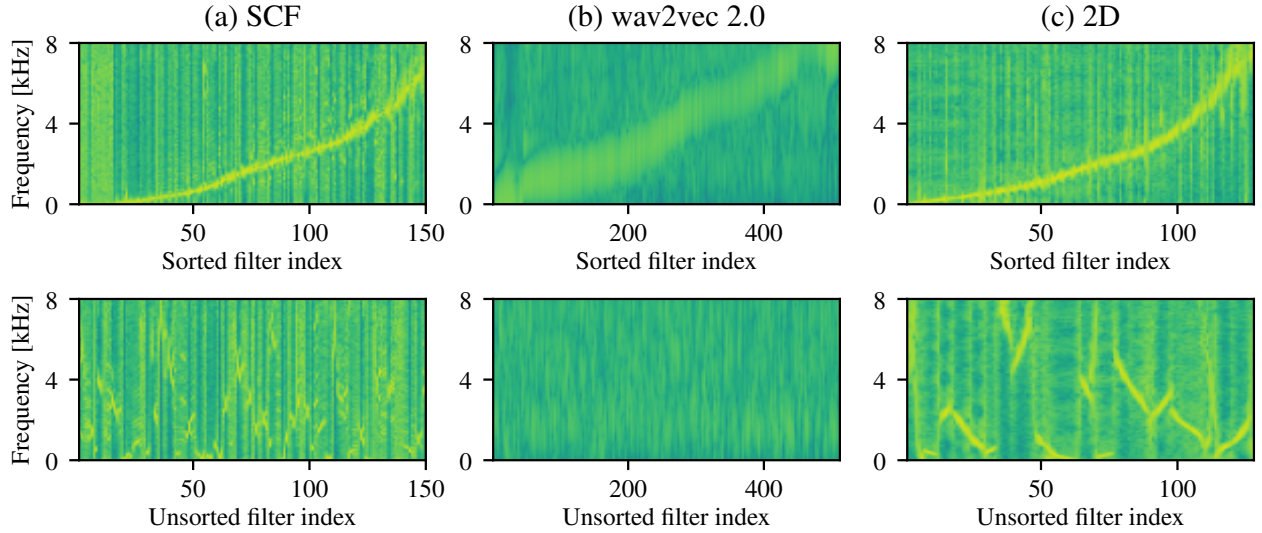


Figure 2: Frequency response of different feature extractors’ filters in the first layer that operates on the waveform. The frequency responses are sorted by the peak frequency in the top row and in the order that the network learned them in the bottom row. For the proposed 2D convolutional front-end, we take the filters of the best-performing configuration in Table 1.

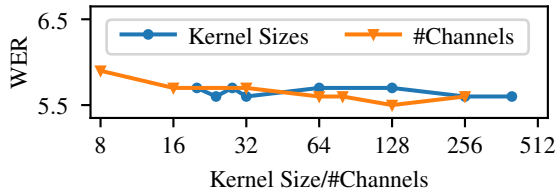


Figure 3: Comparison of different kernel sizes and numbers of channels in the first layer. All experiments use a randomly initialized filterbank with stride 10. For the blue curve, the number of channels is 80 while the kernel size is 256 for the orange curve. After the first layer, 6 2D layers with stride 2 are deployed. Results are WERs on dev-other. The model with kernel size 16 did not converge.

only mildly. Interestingly, this observation is fairly consistent with long-established beliefs in the research field where for Gammatone filters, the minimum length is specified as 32 and the minimum number of channels as 8 [32].

4.1 Analysis of Learned Filters

One way to compare the different filters learned in the first layer is by plotting their frequency responses. The plots can be seen in Figure 2. Given that the order of the filters learned by the neural network is arbitrary in general, we sort the filters by the peak frequency followed by the upper and lower 3 dB cutoff frequencies as second and third sorting criteria in the top row. The frequency responses for SCF and wav2vec feature extractor are consistent with the findings in [11], although it appears that there are fewer unused SCF filters here compared to [11]. Similar to SCF, the first layer filterbank in our proposed generic feature extractor using 2D convolutions also learns a set of bandpass filters whose characteristics are similar to the Gammatone filterbank.

However, a notable difference can be observed when inspecting the unsorted frequency responses in the bottom row of Figure 2. While the filter order appears to be random for SCF and wav2vec feature extractor, the filters learned in the first layer of our proposed feature extractor show clear groups of filters with adjacent ascending or descending center frequencies. This can likely be explained by the nature of the subsequent 2D convolutions with small kernel sizes. When operating with a 3×3 kernel, it can be advantageous for the 2D layers to have related information such as similar frequency bands in neighboring feature channels since only then those channels are seen together by the kernel.

5 Limitations and Future Work

While this paper demonstrates the feasibility of a unified generic front-end to extract features for ASR, the performance of the learnable front-ends falls short of log Mel features. This could be at least partially caused by a lack of audio perturbation. We only use speed perturbation with factors sampled from the fixed set $\{0.9, 1.0, 1.1\}$ which might not have sufficiently strong regularization effect. In contrast, [23] showed improvements when using tempo perturbation with stronger perturbation factors sampled from a continuous distribution. However, tuning the audio perturbation is out of the scope of this work. Another reason might be the low tuning effort for SpecAugment compared to the log Mel baseline while various other aspects of the pipeline such as the learning rate schedule are not tuned at all. Nevertheless, the proposed generic architecture and the insights gained in this work open up many possibilities for further studies on features trained from scratch, possibly also for other tasks besides ASR.

6 Conclusions

In this work, we present a unified generic feature extraction architecture for ASR based on 2D convolutions. The design aims to minimize the influence of traditional feature extraction methods and to unify previously existing architectural inconsistencies in the convolutional front-end. The results using a CTC ASR system on LibriSpeech prove that the proposed architecture is feasible and the remaining difference to log Mel features is not due to its missing structure. This is evidenced by the competitive performance with existing learnable feature extractors. The ablations show that both the STFT and a learnable filterbank can be used for time frequency decomposition in the first layer with similar performance. Furthermore, the front-end is highly parameter-efficient especially for a filterbank with a low number of channels in the first layer, allowing to trade off training speed vs. WER. Finally, our analyses demonstrate that even with the generic design, the front-end exhibits behaviors that align with well-established observations in the field.

7 Acknowledgements

This work was partially supported by NeuroSys, which as part of the initiative “Clusters4Future” is funded by the Federal Ministry of Education and Research BMBF (03ZU2106DD), and by the project RESCALE within the program *AI Lighthouse Projects for the Environment, Climate, Nature and Resources* funded by the Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (BMUV), funding IDs: 67KI32006A.

References

- [1] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” Preprint arXiv:2005.08100, 2020.
- [2] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.
- [3] R. Schlüter, I. Bezrukov, H. Wagner, and H. Ney, “Gamma-tone features and feature combination for large vocabulary speech recognition,” in *Proc. ICASSP*, Honolulu, HI, USA, Apr. 2007, pp. 649–652.
- [4] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *Proc. ICASSP*, Brisbane, Australia, Apr. 2015, pp. 4580–4584.
- [5] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, “Learning the speech front-end with raw waveform CLDNNs,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [6] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, “Learning filterbanks from raw speech for phone recognition,” in *Proc. ICASSP*, Calgary, Canada, Apr. 2018, pp. 5509–5513.
- [7] Z. Tüske, R. Schlüter, and H. Ney, “Acoustic modeling of speech waveform based on multi-resolution, neural network signal processing,” in *Proc. ICASSP*, Calgary, Canada, Apr. 2018, pp. 4859–4863.
- [8] N. Zeghidour, O. Teboul, F. de Chaumont Quirry, and M. Tagliasacchi, “LEAF: A learnable frontend for audio classification,” in *Proc. ICLR*, Vienna, Austria, May 2021.
- [9] M. Ravanelli and Y. Bengio, “Speaker recognition from raw waveform with SincNet,” in *Proc. SLT*, Athens, Greece, Dec. 2018, pp. 1021–1028.
- [10] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, “Acoustic modeling with deep neural networks using raw time signal for LVCSR,” in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 890–894, ISCA Best Student Paper Award.
- [11] P. Vieting, R. Schlüter, and H. Ney, “Comparative analysis of the wav2vec 2.0 feature extractor,” in *ITG Conference on Speech Communication*, Aachen, Germany, Sep. 2023, pp. 131–135.
- [12] B. Kollmeier, T. Brand, and B. Meyer, “Perception of speech and sound,” *Springer handbook of speech processing*, pp. 61–82, 2008.
- [13] H. Hermansky, “Perceptual linear predictive (PLP) analysis of speech,” *The Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.
- [14] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *Computational and Biological Learning Society*, 2015, pp. 1–14.
- [15] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, “Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM,” in *Proc. Interspeech*, 2017, pp. 949–953.
- [16] Y. Zhang, W. Chan, and N. Jaitly, “Very deep convolutional networks for end-to-end speech recognition,” in *Proc. ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 4845–4849.
- [17] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, “ESPnet: End-to-end speech processing toolkit,” in *Proc. Interspeech*, Hyderabad, India, Sep. 2018, pp. 2207–2211.
- [18] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised pre-training for speech recognition,” in *Proc. Interspeech*, Graz, Austria, Sep. 2019, pp. 3465–3469.
- [19] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *Advances in neural information processing systems*, vol. 33, Dec. 2020, pp. 12 449–12 460.
- [20] P. Vieting, C. Lüscher, W. Michel, R. Schlüter, and H. Ney, “On architectures and training for raw waveform feature extraction in ASR,” in *Proc. ASRU*, Cartagena, Colombia, Dec. 2021, pp. 267–274.
- [21] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, “HuBERT: How much can a bad teacher benefit ASR pre-training?” in *Proc. ICASSP*. Toronto, Canada: IEEE, Jun. 2021, pp. 6533–6537.
- [22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” Graz, Austria, pp. 2613–2617, Sep. 2019.
- [23] P. Vieting, M. Kannen, B. Hilmes, R. Schlüter, and H. Ney, “Regularizing learnable feature extraction for automatic speech recognition,” in *Proc. Interspeech*, Rotterdam, Netherlands, Aug. 2025, to appear.
- [24] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” Preprint arXiv:1607.06450, 2016.
- [25] D. Hendrycks and K. Gimpel, “Gaussian error linear units (GELUs),” Preprint arXiv:1606.08415, 2018.
- [26] W. Zhou, W. Michel, R. Schlüter, and H. Ney, “Efficient training of neural transducer for speech recognition,” in *Proc. Interspeech*, Incheon, Korea, Sep. 2022, pp. 2058–2062.
- [27] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “LibriSpeech: An ASR corpus based on public domain audio books,” in *Proc. ICASSP*, Brisbane, Australia, Apr. 2015, pp. 5206–5210.
- [28] M. Bisani and H. Ney, “Joint-sequence models for grapheme-to-phoneme conversion,” *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.
- [29] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” in *Proc. North American Chapter of the ACL*, New Orleans, LA, USA, Jun. 2018, pp. 464–468.
- [30] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, 2019.
- [31] J. D. Kahn, V. Pratap, T. Likhomanenko, Q. Xu, A. Hannun, J. Cai, P. Tomasello, A. Lee, E. Grave, G. Avidov, B. Steiner, V. Liptchinsky, G. Synnaeve, and R. Collobert, “Flashlight: Enabling innovation in tools for machine learning,” in *Proc. ICML*. Baltimore, MD, USA: PMLR, Jul. 2022, pp. 10 557–10 574.
- [32] R. D. Patterson, I. Nimmo-Smith, J. Holdsworth, P. Rice *et al.*, “An efficient auditory filterbank based on the gamma-tone function,” in *a meeting of the IOC Speech Group on Auditory Modelling at RSRE*, vol. 2, no. 7, 1987.