Lightweight End-to-End Speech Recognition from Raw Audio Data Using Sinc-Convolutions

Ludwig Kürzinger¹, Nicolas Lindæ¹, Palle Klewitz¹, Gerhard Rigoll¹
¹Technische Universität München
ludwig.kuerzinger@tum.de, nicolas.lindae@tum.de, palle.klewitz@tum.de

Abstract

Many end-to-end Automatic Speech Recognition (ASR) systems still rely on pre-processed frequency-domain features that are handcrafted to emulate the human hearing. Our work is motivated by recent advances in integrated learnable feature extraction. For this, we propose Lightweight Sinc-Convolutions (LSC) that integrate Sinc-convolutions with depthwise convolutions as a low-parameter machine-learnable feature extraction for end-to-end ASR systems.

We integrated LSC into the hybrid CTC/attention architecture for evaluation. The resulting end-to-end model shows smooth convergence behaviour that is further improved by applying SpecAugment in the time domain. We also discuss filter-level improvements, such as using log-compression as activation function. Our model achieves a word error rate of 10.7% on the TEDDiium v2 test dataset, surpassing the corresponding architecture with log-mel filterbank features by an absolute 1.9%, but only has 21% of its model size.

Index Terms: Sinc convolutions, speech recognition, attention-based neural networks

1. Introduction

End-to-end Automatic Speech Recognition (ASR) systems directly infer tokens (letters or word pieces); sentence transcription is done in a single stage without intermediate representations, e.g., [1, 2, 3, 4]. This structure is desirable, as transcriptions can be directly used as a training objective without having to rely on hard-coded feature extraction or intermediary objectives, such as separately trained acoustic models. However, one may argue that many architectures are not yet fully end-to-end as the encoder still requires preprocessed audio features in the form of log-mel filterbanks. Recent advances using parametrizable Sinc-convolutions enable the integration of the feature extraction as learnable part into the neural network [5, 6, 7].

In this work, we propose a lightweight machine-learnable feature extraction integrated into the ASR neural network to directly classify sentences from raw audio. For this, we extend the established hybrid CTC/Attention system [1] architecture, that combines two main techniques for end-to-end ASR: Neural networks trained with Connectionist Temporal Classification (CTC) [8] calculate the probability of each token at a given time step in the input sequence. Attention-based encoder-decoder architectures [9] are trained as sequence-generating models. The encoder transforms the input sequence into a latent representation. From this, the decoder autoregressively generates the sentence transcription.

Our work is motivated by recent developments in speech and speaker recognition on learnable feature extraction that can be integrated into an end-to-end trainable neural net [5, 6, 7, 10, 16, 17, 18, 19]. Ravanel et al. demonstrated that parametrizable convolutions with a filter structure have better convergence behavior than plain CNNs alone on raw audio data and proposed the SincNet architecture [5] that was first used in speaker recognition [5, 6] and then phoneme recognition [7].

2. Related Work

Recent approaches in end-to-end speech recognition started to outperform conventional hybrid DNN/HMM speech recognition [1, 2, 3, 4, 12]. However, most of these end-to-end systems, such as [1], still rely on preprocessed log-mel f-bank features. A common approach is also to classify using similar methods as in image recognition. For example, Hannun et al. [3] train an end-to-end recurrent network on spectrogram features. But only few architectures directly classify on raw audio [4].

There are two dominating techniques for end-to-end speech recognition: (1) Connectionist Temporal Classification (CTC) [8] carries the concept of hidden Markov states over to end-to-end neural networks as training loss for sequence-classification networks. Neural networks trained with CTC loss calculate the probability of each letter at a given time step in the input sequence. (2) Attention-based encoder-decoder architectures [9] are trained as sequence-generating models. The encoder transforms the input sequence into a latent representation. From this, the decoder autoregressively generates the sentence transcription.

Attention-based sequence transcription was proposed in the field of machine language translation in [13] and applied to speech recognition in [9]. Our work builds on the hybrid CTC/Attention ASR system that combines CTC with the location-aware attention mechanism, as proposed in [14]. Sentence transcription is performed with the help of a RNN language model (RNNLM) integrated into decoding process using shallow fusion [15].

Our work is motivated by recent developments in speech and speaker recognition on learnable feature extraction that can be integrated into an end-to-end trainable neural net [5, 6, 7, 10, 16, 17, 18, 19]. Ravanel et al. demonstrated that parametrizable convolutions with a filter structure have better convergence behavior than plain CNNs alone on raw audio data and proposed the SincNet architecture [5] that was first used in speaker recognition [5, 6] and then phoneme recognition [7].
Net architecture into the hybrid CTC/Attention architecture [1].

Sinc-convolutions were also successfully applied to low-parameter keyword spotting [10] to optimize energy consumption in battery-powered devices. The model proposed by Mittermaier et al. employs depthwise separable convolutions [20, 21] on top of Sinc-convolutions. Depthwise separable convolutions are useful in low-parameter scenarios [22, 23]. A similar approach to reduce the number of parameters in convolutions are lightweight convolutions [24] that are based on depthwise convolutions.

In our proposed method, we employ depthwise convolutions that omit the pointwise convolution of depthwise separable convolutions [20].

3. Architecture

This section describes the proposed feature extraction block as our main contribution and, in a second part, its integration into an existing end-to-end ASR system. This architecture directly infers token and sequence probabilities from raw audio, without any previously extracted log-mel features.

3.1. Front-End: Lightweight Sinc-Convolutions (LSC)

To obtain higher-level latent representations from a sequence of raw audio frames \( X = x_{1:T} \), we build upon one layer of parametrized Sinc-convolution and several layers of depthwise convolutions (DConv). These layers extract filter features \( R \) from our input, replacing pre-calculated log-mel f-bank features that are commonly used in end-to-end ASR systems:

\[
R = r_{1:T} = [\text{DConvBlock}]^3(\text{SincBlock}(x_{1:T}))
\]  

We term this architecture Lightweight Sinc-Convolutions (LSC), as shown in Fig. 1. Based on the Sinc function \( \text{sinc}(x) = \sin(x)/x \), the Sinc-convolution is constructed from the convolutional kernel

\[
g[n, f_1, f_2] = 2f_2 \text{sinc}(2\pi f_2 n) - 2f_1 \text{sinc}(2\pi f_1 n),
\]

that acts as a bandpass filter in the spectral domain with \( f_1 \) and \( f_2 \) as the cutoff frequencies. Unlike with pre-computed filterbank features, the above formula can be integrated into a neural network as a layer that is differentiable w.r.t. the cutoff frequencies. To better approximate an ideal Sinc filter, the Hamming window \( w[n] = 0.54 - 0.46 \cos(2\pi n/L_s) \) is multiplied with the \( L_s \) weights in the Sinc-convolution kernel. The cutoff frequencies of a bandpass filter are derived from the learnable parameters \( w_1 \) and \( w_2 \) [5]:

\[
f_1 = |w_1|, \quad f_2 = |w_1| + |w_2 - w_1|
\]

For non-linearity, we use log-compression as activation function of this layer, defined as \( \text{LogCompression}(x) = \log(|x| + 1) \). In comparison with the ReLU activation function as used in SincNet [5], log-compression drops less information and can improve the classification performance on raw audio [2, 10, 25, 26].

The Sinc-convolutional layer is followed by multiple depthwise convolutions (DConvs) for extraction of higher-level latent representations. DConvs use separate kernels for each input channel, reducing the number of parameters of a DConv layer with \( c_{in} \) input and \( c_{out} = n \cdot c_{in} \), \( n \in \mathbb{N} \) output channels and a kernel of size \( k \) from \( c_{in} \cdot c_{out} \cdot k \) to \( c_{out} \cdot k \). They extract short-time context along the time dimension, however, limited to the data within one frame. This part of the architecture acts as coupling layer between the Sinc-convoluted signal and the ASR back-end network. LSC are heavily inspired by the lightweight keyword spotting architecture by Mittermaier et al. [10] using depthwise separable convolutions. Our adaption mainly omits the inter-frame pointwise convolution in order to further reduce the number of parameters. DConvs contribute more to the extraction of high-level features than pointwise convolutions, while making up only a small fraction of the learnable parameters [27].

3.2. Back-End ASR Architecture

The back-end architecture transforms the extracted features provided by the LSC front-end into a token sequence, see Fig. 2. The first part, the encoder, consists of BLSTM layers with projection neurons (BLSTMP) as proposed in [1, 12]. It maps the sequence of latent representations \( R = r_{1:T} \) to a sequence of high-level representations, i.e., \( \hat{h} = h_{1:T} = \text{BLSTMP}(x_{1:T}) \).

On top of the encoder, one layer maps the hidden state sequence \( h_{1:T} \) to the predicted output sequence of tokens \( \hat{y}_{1:T} \) of shorter or equal length \( L_1 \). This layer is trained with Connectionist Temporal Classification (CTC) loss. For this, we first use a linear layer with softmax activation to map the hidden state sequence to a sequence of token probabilities, where the list of tokens has been extended with a blank token: \( \hat{y}_{1:T} = \text{softmax}(\text{Linear}(h_{1:T})) \). Each \( \hat{y}_t \) is a probability distribution over the tokens for timestep \( t \). From these, the probability \( p_{\text{CTC}}(Y|X) \) of our target sequence \( Y = y_{1:L} \) is calculated by summing up the probabilities of all possible paths through \( \hat{y}_t \) leading to the target sequence, where we remove duplicate symbols that aren’t separated by a blank token. The CTC loss is the cross entropy of that probability:

\[
L_{\text{CTC}} = -\log p_{\text{CTC}}(Y|X).
\]

For beam search decoding, the prediction \( \hat{y}_{1:T} \) for the most probable sequence is obtained using a modified forward algorithm [1].

The hybrid CTC/Attention architecture incorporates a LSTM based decoder network that utilizes location-aware attention as a method of attending arbitrary encoder states and thereby eliminating the bottleneck between encoder and decoder. At step \( l \), location aware attention retrieves arbitrary encoder states by computing an importance score

\[\text{score}(h_t, q_{l-1})\]
from the language model and estimate the most probable target sequence from that using beam search, as shown at the bottom of Fig. 2.

4. Experiments

We evaluate our architecture on the freely available TEDlium v2 corpus [28] with more than 200 hours of speech. It is a complex ASR task widely used for speech recognition. The raw audio stream is split into frames of 25 ms of raw audio, with two consecutive frames overlapping by 15 ms, resulting in a stride of 10 ms\(^1\). As the audio was recorded with a sampling frequency of 16kHz, this means our frames consist of 400 samples each, the overlapping section of 240 samples. We trained our network using the Adadelta optimizer (\(\rho = 0.95, \epsilon = 10^{-8}\)) for 22 epochs before early stopping ended training, and then chose the model with the highest validation accuracy for evaluation. The Sinc convolution consists of 128 Sinc filters initialized with mel filterbank weights. For our network’s encoder, we chose to use four BLSTM layers with projection neurons of size 512. The decoder is a single LSTM layer of size 512 in combination with an attention layer also of size 512. We chose 0.5 as our CTC/Attention weight and trained with 500 unigram units as tokens, created by SentencePiece [30].

As additional data augmentation, we add spectral augmentation [31] to the training procedure. Spectral augmentation introduces two augmentations techniques: Time warping distorts part of the input features, and the dropout-like masking is applied to blocks in both feature and time dimension. Although this technique is meant to be used on already extracted filter features, we found it to improve ASR performance when applied on a framed audio signal. We used the standard configuration as provided by the ESPNet toolkit, although it is sensible to further investigate and adjust these values in the future. For decoding, we used a pre-trained language model provided by ESPNet. The language model was integrated with a fusion weight of \(\beta = 0.5\), the CTC weight was chosen to be \(\lambda = 0.4\).

5. Results and Discussion

As a baseline model to compare to, we use the hybrid CTC/attention ASR as proposed in [1, 12]. It also uses the location-aware attention mechanism, but classifies from logmel FBank features. We compare our model performance to the best published VGG-BLSTM result [12]\(^2\). An overview that shows parameters of our architecture compared to this model from ESPNet is given in Table 1. Our network is smaller in the number of learnable parameters compared to VGG-BLSTM. We managed to create a front-end with fewer parameters due to Sinc and depthwise convolutions: LSC uses only 16k parameters, whereas the VGG layers learn 259k parameters. Our low parameter count can be attributed to depthwise convolutions. In the proposed parameter configuration as seen in Fig. 1, adding the pointwise convolution in the DConv blocks would result in additional 240k parameters. We were also able to use thinner back-end layers of size 512 (instead of 1024 used in the configuration of the VGG-BLSTM model), resulting in only 22.3m parameters instead of 106m. The CTC layer requires an architecture that provides a short-term context; here, it’s provided by

\(^{1}\) Frame parameters were chosen according to the standard setting for MFCC features in the kaldi toolkit [29].

\(^{2}\) Results obtained from the best VGG-BLSTM model as listed in the ESPNet github repository at commit hash f36bbee

Figure 2: ASR Back-end: Features extracted by the front-end are passed through a BLSTM encoder. Encoder states are used by a CTC and an attention decoder to decode the most probable sequence. The decoding process is guided by a beam search.
the BLSTM in the encoder, so that frame-wise 1D filter convolutions can be used without losing inter-frame information.

Table 1: Architecture comparison between the hybrid CTC/attention VGG-BLSTMP model [1] and our LSC+BLSTMP architecture.

<table>
<thead>
<tr>
<th></th>
<th>VGG-BLSTMP (ESPnet)</th>
<th>LSC-BLSTMP (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>F-bank+pitch</td>
<td>Sinc-Convs</td>
</tr>
<tr>
<td>Window</td>
<td>Hann [32]</td>
<td>Hamming</td>
</tr>
<tr>
<td>Frame size</td>
<td>25 ms</td>
<td>25 ms</td>
</tr>
<tr>
<td>Frame shift</td>
<td>10 ms</td>
<td>10 ms</td>
</tr>
<tr>
<td>Freq. bins</td>
<td>80+3</td>
<td>128</td>
</tr>
<tr>
<td>Coupling</td>
<td>VGGNet</td>
<td>DConvs</td>
</tr>
<tr>
<td>Output dim</td>
<td>2688</td>
<td>256</td>
</tr>
<tr>
<td>Size front-end</td>
<td>259k</td>
<td>16k</td>
</tr>
<tr>
<td>Size back-end</td>
<td>106m</td>
<td>22.3m</td>
</tr>
</tbody>
</table>

Table 2: Comparison of performance on the TEDlium 2 dataset [28]. Listed are the best VGG-BLSTMP WER from ESPnet [12], of a VGG-BLSTMP with a to our model comparable network size; and the variations of our architecture with LSC.

<table>
<thead>
<tr>
<th>Model</th>
<th>Fea.</th>
<th>LM</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-BLSTMP [1]</td>
<td>FBANK</td>
<td>✓</td>
<td>12.8</td>
<td>12.6</td>
</tr>
<tr>
<td>VGG-BLSTMP (small)</td>
<td>FBANK</td>
<td>✓</td>
<td>19.8</td>
<td>18.6</td>
</tr>
<tr>
<td>LSC</td>
<td>RAW</td>
<td>✓</td>
<td>15.1</td>
<td>15.7</td>
</tr>
<tr>
<td>LSC+SpecAug</td>
<td>RAW</td>
<td>✓</td>
<td>13.5</td>
<td>13.4</td>
</tr>
<tr>
<td>LSC+SpecAug (ReLU)</td>
<td>RAW</td>
<td>✓</td>
<td>11.4</td>
<td>11.6</td>
</tr>
<tr>
<td>LSC+SpecAug (logCpr)</td>
<td>RAW</td>
<td>✓</td>
<td>10.7</td>
<td>10.7</td>
</tr>
</tbody>
</table>

Despite its smaller size, the model still achieves a better WER performance on the TEDlium v2 test set. By inspection, a smooth asymptote-like of model loss convergence can be observed during training on this dataset. Table 2 shows performance comparisons of our architecture on the TEDlium 2 dataset with the VGG-BLSTMP model. We achieve a 0.9% lower WER of 11.7% on the test set when decoding with a language model of similar size (8.9m compared to 7.4m parameters). When using the large language model used for ESPnet’s transformer network, we can further reduce the WER by 1% to 10.7%, however, the parameter count of the larger LM increases to 139m parameters. Although we use a large RNNLM for evaluation, a more lightweight LM can be implemented depending on application scenario. To provide a more distinctive and fair comparison, we also list a parameter-reduced VGG-BLSTMP model with only 19m parameters that is comparable in model size to our evaluated model. This model however only achieved a WER of 18.6% [33].

Training with spectral augmentation also improved the network’s performance, despite not fine tuning the configuration for raw audio data as input: Spectral augmentation improved the WER by 2.3% and 0.9% when decoding without and with the large language model. To evaluate log-compression as alternative activation, we also tested one model instance using ReLU as activation function in the SincBlock (as in Fig. 1). With ReLU, we achieved slightly worse WERs of 11.6%/10.9% for the dev/test set, confirming the results from Mittermaier et al. [10, 25]. This indicates that log-compression is beneficial to the extraction of filter features from raw audio.

Fig. 3 shows four learned Sinc-convolution filters of the LSC, and Fig. 4 visualizes the learned filters by sorting them and plotting their upper and lower bounds. In those filters, a trend towards a higher amplitude is noticeable, also towards a wider band pass in the spectral domain. Notably, one of the Sinc filters converged to pass through the entire raw audio signal, indicating an inclination of the network to directly process the raw data.

6. Conclusions

Many end-to-end ASR systems still rely on pre-processed f-bank features. Our work extended an existing ASR system with a front-end to classify directly from raw audio that we termed lightweight Sinc-convolutions (LSC). Its network architecture is carefully chosen to be lightweight, i.e. using less parameters, which is an advantage for speech recognition in battery-driven devices. We discussed further improvements to this architecture, such as log-compression as activation function and spectral augmentation for time-based data augmentation. Our final model achieves a word error rate of 10.7% when combined with a large RNN language model, an improvement of absolute 1.9% over the best reported model of the corresponding f-bank architecture, but only has 21% of its model size.
7. References


