Controlling the Noise Robustness of End-to-End Automatic Speech Recognition Systems

Matthias Möller, Johannes Twiefel, Cornelius Weber, Stefan Wermter
Knowledge Technology, Department of Informatics
University of Hamburg, Germany
Hamburg, Germany
{7moeller, twiefel, weber, wermter}@informatik.uni-hamburg.de

***Abstract*—In this work, we propose a novel training scheme to modularize end-to-end systems. Our training scheme aims at altering the flow of information in an end-to-end system to use the kernels of this system for another system that fulfills another task. We apply this scheme to extract the noise reduction capabilities from a noise-robust *automatic speech recognition* (ASR) system and implement a speech enhancer from it. This enhancer receives spectral representations from unfiltered audio and outputs cleaned spectral representations. Our enhancer can be integrated into an ASR system as front-end, is trainable, and reduces background noise. Our front-end uses a decoder to clean speech based on the hidden activations of the ASR system Jasper. While training, we exclusively adapt the weights in our decoder and the batch normalization in Jasper. The resulting spectral representations show less background noise. Further, areas in the spectral features are not reconstructed if they do not contribute to speech recognition. We demonstrate that our front-end can be combined with a pre-trained ASR system as back-end and supports speech recognition in noisy conditions. Further, we show that training another ASR system with our front-end results in an increased performance of the ASR system in noisy as well as noiseless conditions. The ASR system's performance is especially improved on challenging speech datasets.**

I. Introduction

One of the most prevalent approaches to increase the performance of speech recognition systems is speech enhancement which deals with enhancing the presence of speech in sound while diminishing noise. A problem results from the aspect that one cannot easily quantify noise in speech. Several metrics that measure the physical characteristic of noise do not necessarily correlate with the perceptual presence of noise. Psychoacoustics has shown that the human perception of sound and noise is influenced by aspects like the frequency or the amplitude of the respective speech signal [1], [2]. Many metrics of speech enhancement are designed based on human hearing through metrics like *mean opinion score* (MOS). However, those metrics have shown to be an inadequate way for measuring from the perspective of ASR systems [3]. Consequently, several algorithms that were designed to optimize according to these metrics show questionable results for ASR systems [3].

On the other hand, pre-trained, noise-robust ASR systems have learned to deal with different sources of noise. Implementing the noise reduction capabilities of such systems into a separate speech enhancer could have many sustainable benefits. The module could support other pre-trained and less noise robust ASR systems in noisy environments. That would be attractive especially for less noise robust ASR systems like DeepSpeech [3], [4]. Such an enhancer can thereby contribute to the application of other already trained ASR systems in real-world scenarios. Further, the enhancer could also support the other ASR systems in training. ASR systems that are training on the output of the enhancer would not need to learn to process the removed noise which can lead to a decrease in resources for training and result in smaller ASR systems. Furthermore, we address the possibility that the enhancer could also be supportive for other speech-related tasks that provide fewer data than English ASR. This involves ASR of other languages but also tasks in the field of emotion recognition or speaker identification.

However, most ASR systems are *End-to-End* (E2E) systems. These are usually large monolithic structures that cannot easily be modularized. For isolating the noise reduction capabilities from an ASR system, one has to address the questions if and how the internal filtering mechanisms can be accessed. *Li et al.* [5] address these questions in their work, successfully reconstructing spectral representations of speech by the hidden activations of an ASR. Their reconstructed spectral representations show that background noise and speaker information were less reconstructed with an increasing number of reused layers of the ASR systems.

Our work takes inspiration from the work of *Li et al.* [5] but extends their approach with new ideas. Our speech enhancer also reconstructs spectral representations from an ASR system. However, we additionally modulate the flow of information in the ASR system to support the reconstruction of clean and natural spectral representations. For this purpose, we use the optimization technique *batch normalization* (BN) to let a network learn which kernels in a noise-robust ASR system contribute to noise reduction. We combine this idea with a noise-robust ASR system that provides many possibilities to forward information by its architecture. The result is a trainable front-end that removes background noise, retains speaker information but also deletes frequencies that do not contribute to speech recognition concerning the speaker.

This work was supported by the German Research Foundation (DFG) under project "Crossmodal Learning" (TRR-169).

II. Related Work

A. Dataset: LibriSpeech

LibriSpeech [6] is an English speech corpus that contains about 1000 hours of speech. *Panayotov et al.* [6] derive this corpus from audiobooks. LibriSpeech is freely accessible and contains speech data that were sampled at 16 kHz. It is a widely used dataset for training ASR systems [7]–[9], represents the standard dataset for evaluating ASR systems and reporting benchmark results in English.

LibriSpeech is divided into training, test, and validation set. The validation set is also called the development set. Each set is divided into at least two subsets, which differ because the respective speakers originate from two different pools of speakers. These pools were created by ranking the speakers according to a *word error rate* (WER) they obtained by applying another ASR system. The "clean" pool contains the half with the better performing speakers, while the "other" pool contains the rest. Thus, both speaker pools do not share speakers.

The subsets "dev-clean" and "test-clean" consist of 20 male and 20 female speakers who were randomly chosen from the "clean" pool. That results in approximately 8 minutes of speech for each speaker. The remaining speakers in the "clean" pool were used to create the "train-clean-100" and the "train-clean-360" set. The "other" pool was used similarly to generate a development and test set. First, the speakers in the "other" pool were ranked according to the WER, and four sub-pools were created based on the speaker's rank. The speakers for the test and development set were randomly chosen from the third sub-pool. The remaining speakers were used to generate the "train-500-other" set.

B. Dataset: Noisy Speech Database

The *Noisy Speech Database* (NSD) [10] is a dataset that was designed for training and testing speech enhancement algorithms. It is divided into a train- and a test set. This dataset contains clean and noisy speech signals with transcriptions for the spoken sentences. Thus, it represents a baseline for evaluating ASR systems as well as speech enhancement algorithms.

The training set is divided into a 28- and a 56-speakers dataset [11], [12]. The speakers in the 28-speakers dataset share the same British accent, while the 56-speakers dataset contains a wider variety of accents [12]. Each training set contains the same number of male and female speakers. In both datasets, each speaker has about 400 sentences of speech [11]. The samples of clean speech for each speaker were received from the Voice Bank corpus [13].

For generating noisy speech samples, eight sources of noise were received from the DEMAND database [14], and two other sources of noise were artificially generated. The artificially generated sources of noise were created by using real speech. One noise was created by adding the voices of six different speakers into one sound and the second noise consisted of white noise that was adapted to have a frequency response that is similar to the voice of a male speaker. This noise was added to the clean speech with different *sound-to-noise ratios* (SNRs). A lower SNR indicates an increased influence of the noise on the speech signal. For both training sets, the following SNRs were chosen: 0, 5, 10, 15 dB. Thus, 40 different noise conditions (10 types of background noise $\times$ 4 SNRs) were generated. The noise conditions were equally distributed on all 400 sentences for each speaker. That resolves in 10 sentences per speaker for each noise condition.

The test set was constructed in the same way as both training sets. It differs in selected speakers, types of background noise, and SNRs. The test set was generated by receiving approximately 400 speech data of one male and one female speaker each. As background noise, five different noises were obtained from the DEMAND database. As for the SNR, 2.5, 7.5, 12.5, 17.5 dB were chosen as respective values [12]. This combination resulted in about 20 sentences per speaker for each noise condition.

C. Batch Normalization

Batch normalization (BN) [15] was introduced to counter the internal covariate shift. This internal covariate shift describes the changes in the output distribution of a network's hidden by adapting its parameters through training [15]. The changes in the output of a hidden layer are the input for the subsequent layer and have negatively impact training time. A BN layer aims to fix the output distribution of a hidden layer by z-score normalization. However, z-score normalization would influence what a layer can present and learn [15]. Therefore, a BN layer applies two trainable parameters: a scaling factor γ and a bias β:

$$BN(x) = \gamma \cdot \hat{x} + \beta \qquad \hat{x} = \frac{x^{(k)} - E[x^k]}{\sqrt{Var[x^{(k)}]}}$$

By using these parameters, a BN layer can learn to invert the z-score normalization by adapting $\gamma = \sqrt{Var[x]}$ and $\beta = E[x]$.

D. Convolutional Neural Networks in Speech

An ASR system can often be divided into three components: a front-end, an acoustic model, and a decoder. The front-end's task is the conversion from speech signals into sequences of spectral features which in the context of ASR are conventionally called speech features. These features are calculated for a chosen time frame and overlap from the original wave. A popular choice for a time-interval and a time-overlap are 20ms with 50% overlap. In practice, this choice results in about 100 spectral features per second of speech. The acoustic model outputs probability distributions over characters for every speech feature. That results in a sequence of 100 predictions per second. The decoder's task is the conversion of these probability distributions over characters into words.

In the last years, several ASR systems emerged that reach benchmark or near-benchmark results by mainly or entirely using 1D *convolutional neural networks* (CNN) in their acoustic model [7], [8], [16]. A 1D CNN describes a CNN whose kernel spans all other dimensions completely, apart from the one that is convolved over. The output of a layer is created by shifting

the kernels from left to right. In ASR systems, these kernels span multiple speech vectors completely and only shift along the temporal axis. Usually, a "same padding" [17] is applied to counter the loss of the temporal axis. Thereby, the acoustic model still provides a prediction for each speech feature.

E. Jasper

Jasper [7] is a family of ASR systems that are characterized through the use of BN, 1D CNN, ReLu, Dropout, and skip connections. Several near-benchmark and benchmark results were reported for this family.

Jasper networks are conventionally structured into sub-blocks and blocks. A sub-block describes a sequence of 1D CNN, BN, ReLU, and Dropout. A block is a sequence of subsequent sub-blocks. Conventionally, Jasper models are named according to their blocks and sub-blocks: a model that is called *Jasper* $B \times R$ refers to an architecture that consists of B blocks where each block consists of R sub-blocks.

Each Jasper model can be applied in two different topologies namely the Jasper $B \times R$ and the Jasper *dense residual* (DR) $B \times R$. Both topologies differ only in how they apply residual connections. The Jasper $B \times R$ topology describes an architecture where the input of each block is forwarded into the output of its last sub-block. The Jasper DR $B \times R$ topology describes an architecture that forwards the input of each block into the last sub-block of each block.

The best performing representative of the Jasper family is the Jasper DR 10×5 model [7]. The best model was trained with a data augmentation technique and an optimization technique called NovoGrad [7], [18].

F. Reconstructing Spectrograms based on a Network's Hidden Activations

Li et al. [5] presented an approach to analyzing ASR systems. Their approach consisted of reconstructing an ASR systems' input from its hidden activations. Therefore, decoder networks were injected at the hidden activations of several ASR systems. Only the decoder network's parameters were adapted while training. The decoder structure is referred to as the probing model or reconstruction model. All probing models were 4-layered highway networks [19].

Several probing models were injected in several layers of four different ASR systems. Those ASR systems can be divided into two different network architectures that were trained either with or without a data augmentation technique. The first architecture was a pure-LSTM that consisted of five bi-directional LSTM layers. The second architecture was a VGG-LSTM that consisted of 4 convolutional layers and 5 subsequent LSTM layers. All ASR systems were trained on the 100-hour subset of LibriSpeech.

Afterwards, the reconstructed spectrograms were analyzed according to several speech enhancement metrics and individual listening. Their overall results imply that the first layers reduced background noise. However, speaker information we also not reconstructed and the reconstructed speech was less natural but sounded artificial.

III. Approach

Our approach aims at isolating the noise reduction capabilities of an E2E ASR system and uses these as part of the implementation of a speech enhancer module. We take inspiration from the research of *Li et al.* [5]. They proposed an approach to analyzing the processing mechanisms of ASR systems by reconstructing spectrograms from their hidden activations. Their reconstructed spectrograms showed less background noise with the shortcomings of losing speaker information. Thus, we are presented with an approach to access the noise reduction capabilities of ASR systems. However, the loss of speaker information contradicts the task of a speech enhancer that enhances speech while diminishing noise. Speech that is too synthetic can deteriorate the performance of ASR systems that have trained with clean data [20].

We address the question of whether we can reconstruct spectral representations that show less background noise without the shortcomings of losing speaker information. In other words, we address the question of whether we can construct a speech enhancer that cleans speech based on an ASR system. Our speech enhancer produces spectral representations and thereby fulfills the role of a front-end in an ASR system. The construction of our front-end follows the implementation of *Li et al.* [5] with the following distinctions:

1) we reconstruct spectral representations from the hidden units of Jasper,
2) we train the BN in Jasper, and
3) we use a speech enhancement dataset for training.

We expect that these distinctions will lead to reconstructed spectrograms in the upper layers that contain less background noise while retaining speaker information. We argue as follows: If Jasper has sequences of kernels that contribute to noise reduction, forwarding information mainly through those could support the reconstruction of cleaned spectral representations. Advantageously, Jasper applies by its architecture many skipping connections that allow forwarding information more freely. Additionally, Jasper implements many BN layers with two trainable parameters. By training both parameters, a BN layer can learn to counter the z-score normalization [15]. On the other hand, it can also learn to stop the flow of information entirely. Lastly, we consider the training with a speech enhancement dataset as beneficial for letting the network learn to identify noise reduction kernels better.

We see several advantages in our approach: Firstly, we expect that our front-end will learn to ignore many kernels in Jasper that do not contribute to noise reduction. Therefore, we see a high potential in shrinking the front-end afterwards. The shrinking is a task of future work but could result in a much smaller architecture that is faster and applicable for devices that provide less performing hardware. Secondly, we avoid addressing the question of which layer is best suited for reconstructing spectral representations. Our training scheme aims to let the front-end learn which kernels are productive for the reduction of noise throughout the whole ASR system. Thirdly, we expect the deletion of more than background noise.

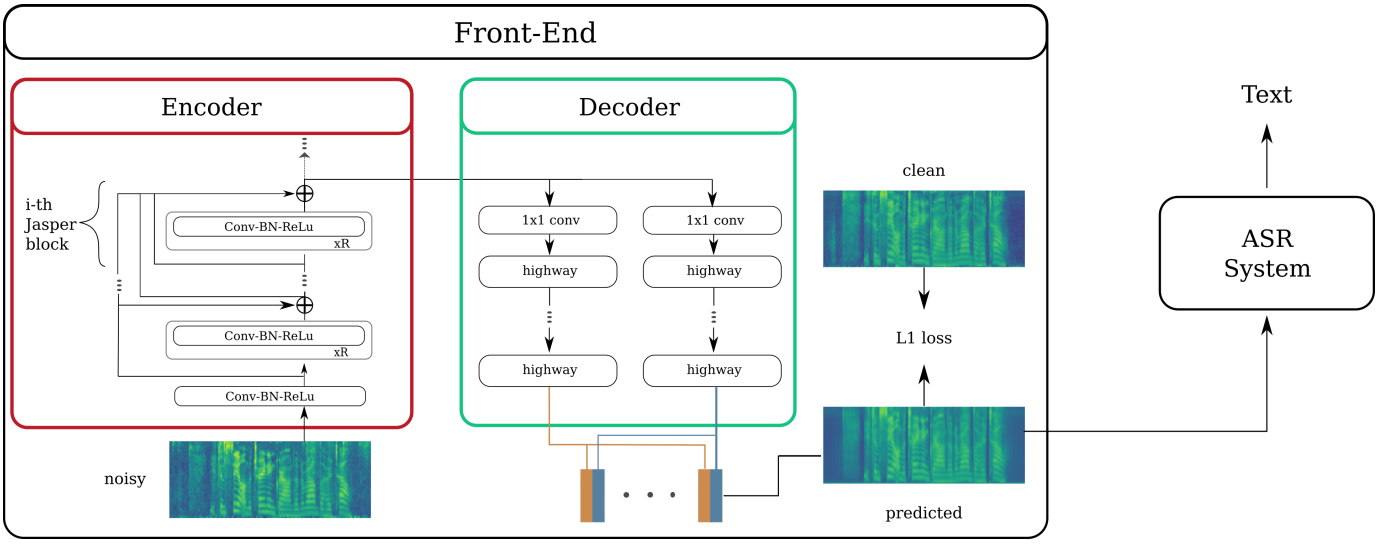

Fig. 1. The front-end receives audio as input and reconstructs normalized log Mel-filterbanks. It consists of an encoder (red outline) and a decoder (green outline). The encoder consists of the reused Jasper blocks which read log Mel-filterbanks. We do not use the Dropout for the training and only train the BN. The sub-networks in our decoder predict one frame from the hidden activations of Jasper each which are shown in orange and blue. The predicted spectrograms can afterwards be fed into an ASR system.

On the other hand, the deleted information could be a result of the inner processing mechanisms of an ASR system. We could receive insight into what noise is from the perspective of an ASR system.

A. Architecture

We divide our front-end into encoder and decoder (cf. Figure 1). The encoder consists of the reused layers of Jasper, while the decoder is the probing model. Our front-end receives spectrograms and predicts cleaner spectrograms of the same format. Jasper uses normalized logarithmized Mel spectrograms. Thus, we restrict our front-end to output cleaner normalized Mel spectrograms.

For our encoder, we reuse the Jasper DR 10×5 model. This model showed superior performance on LibriSpeech and is publicly available. Additionally, we found this architecture more noise-robust than Google ASR. By testing, we observed that Jasper received a WER of 18% on the NSD test set, while Google achieved 23%. Furthermore, the DR topology applies additional residual connections between their blocks [7]. This architecture seems especially beneficial for our training scheme. We restrict our encoders by only reconstructing speech on the hidden activations after the blocks of Jasper. However, every front-end has to use the first block in Jasper. Jasper's first block is a preprocessing block that outputs half as many vectors as it receives speech vectors, which results from the convolutional layer involving a convolutional stride. One of Jasper's last blocks counters this decrease in the size of the temporal dimension, however, this block is removed in our encoder. In our following experiments, we reconstruct speech from different layers of Jasper. That requires removing the last blocks.

Our decoder is provided with half as many activations as needed for the reconstruction of every speech vector. We counter this by applying two separate networks. Both networks first apply a fully connected layer. The output of the fully connected layer is afterwards fed into a sequence of highway layers. These linear projections are necessary because highway layers require the output and input to be of the same size [19]. Thus, we cannot decrease the size along the spectral dimension by the highway network. Our first sub-network is tasked to reconstruct every second frame. Our second sub-network aims to reconstruct every remaining frame.

We train two networks in our decoder simultaneously, avoiding separate training but relying on a gradient-based optimization technique. NovoGrad [18] is related to the family of *stochastic normalized gradient descent* (SNGD) optimizers. Thus, it aims to use the direction of gradients but is less affected by its magnitude. It averages gradients layer-wise. Since our two separate networks have different layers, the actual gradients are calculated separately for each sub-network. Additionally, NovoGrad outperforms other optimizers in their strongest domains while using less memory. Further, we expect the highway layers to support the training of two separate networks. Highway layers apply gating mechanisms that modulate the flow of information. Furthermore, highway networks have been shown to perform well with CNNs [21], [22].

B. Training

We expect that by retraining the γ and the β in the BN, our front-end will learn to use certain kernels in Jasper for cleaning. If a kernel does not contribute to the process of cleaning, our front-end shall learn to avoid it. Thus, we argue that an increased number of reused blocks does not negatively impact the performance of our front-end. To address this statement in our later experiments, we train three different

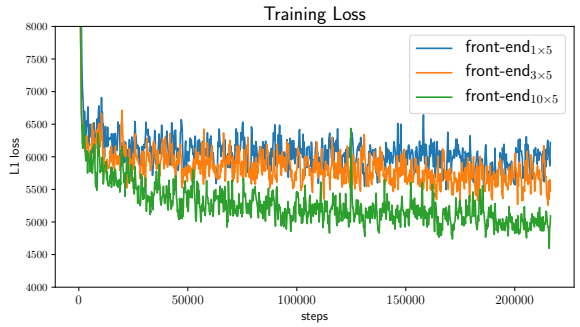

Fig. 2. We train our three front-ends over 400 epochs. The respective graph shows the training loss over the training steps.

front-ends. Each front-end differs in the number of blocks that are reused from the Jasper DR 10×5. We reuse one, three, and eleven blocks. We refer to our front-ends as front-end$_{10\times5}$, front-end$_{3\times5}$, and front-end$_{1\times5}$ respectively.

We train in each configuration only the BN in the encoder and the decoder. We use the same loss used by *Li et al.* [5] and receive most hyperparameters from Jasper's proposed configuration [7]. All front-ends are trained over 400 epochs with a batch size of 64 with the L1 loss. Additionally, we use NovoGrad but deactivate the weight decay. Further, we deactivate Dropout and avoid any regularization. Our decoder uses 8 highway layers and a polynomial learning rate of 0.9. Our biggest front-end - the front-end$_{10\times5}$ - requires 24 hours 13 minutes of training time on a single NVIDIA Titan RTX. The training curves can be seen in Figure 2.

IV. Experiments

A. Experiment I

This experiment addresses the questions 1) of whether our front-ends remove background noise, 2) which information is additionally removed and 3) if our largest architecture, front-end$_{10\times5}$, produces the most natural spectral representations. We assume that by training the BN in the Jasper blocks, our front-end has learned which kernels in Jasper can be used for filtering. Thus, we expect that our biggest front-end performs best.

1) Dataset and Metrics: We choose the test set of NSD for evaluation. The NSD test set provides us with clean speech samples, noisy speech samples, and respective transcriptions. The training and test set in the NSD share neither speakers nor types of noise or SNRs. Thus, our front-ends have not seen any of the tested noise conditions or speakers. Additionally, this dataset provides information about the speaker, location (background noise), and SNR for every sample. This information supports us in identifying the weaknesses of our front-ends.

In this experiment, we use two different metrics. The *mean absolute error* (MAE) enables us to measure the deviation between two spectral representations. Thus, this metric provides us with the possibility to quantify the deviation between our predicted spectrograms to the spectrogram of the actual clean signal. The *word error rate* (WER) is the standard metric for the evaluation concerning ASR systems. We use this metric to gain insight into how well our front-end can be used for another already trained Jasper model. This Jasper model is the Jasper DR 10×5 model that was the baseline for our three front-ends. We speculate that our front-ends - if they are a fair presentation of the capabilities to process noise - shall not interfere with the performance of the original model.

TABLE I
The entries present the calculated MAE of the clean spectrogram to the noisy spectrograms or the spectrograms that are produced by our front-ends. MAE is calculated for different speakers, SNR levels, and background noises (locations). The notation is mean ± standard deviation.

		front-end		
	noisy	1×5	3×5	10×5
speaker				
male	0.48±0.17	0.29±0.07	0.28±0.07	**0.26±0.07**
female	0.48±0.18	0.29±0.07	0.28±0.07	**0.26±0.07**
SNR				
2.5	0.63±0.16	0.34±0.08	0.33±0.07	**0.31±0.08**
7.5	0.52±0.14	0.29±0.06	0.28±0.06	**0.27±0.06**
12.5	0.42±0.12	0.27±0.05	0.26±0.06	**0.24±0.06**
17.5	0.34±0.12	0.24±0.04	0.23±0.04	**0.22±0.04**
background noise (location)				
bus	0.36±0.14	0.25±0.05	0.24±0.06	**0.23±0.06**
cafe	0.61±0.16	0.33±0.08	0.32±0.08	**0.31±0.08**
living	0.56±0.14	0.31±0.07	0.30±0.07	**0.28±0.07**
office	0.36±0.10	0.25±0.04	0.24±0.04	**0.22±0.04**
psquare	0.51±0.14	0.29±0.06	0.28±0.06	**0.26±0.06**

2) MAE on Spectrogram Reconstruction: We start to investigate how far the reconstructed spectrograms deviate from the clean spectrograms. We feed the noisy spectrograms into our three different front-ends and predict new spectrograms. Afterwards, we measure the MAE between clean to noisy and clean to predicted. Our results are shown in Table I. We observe that the MAE between the clean spectrograms and the predicted spectrograms is lower than the MAE between clean spectrograms the noisy spectrograms for all our front-ends. Thus, all our front-ends seem to reduce the deviation to the clean spectrogram. Further, we observe that the MAE between the female and the male speaker is nearly identical in each block for every front-end. Further, we notice that our front-ends reduce the MAE more efficiently if the MAE between noisy and clean is high.

Overall, we consider MAEs between 0.2 and 0.3 as a small deviation for normalized log Mel filterbanks. As we can see in Figure 3, the bins in our spectrograms can have values between -3 and 4 in some samples. Thus, an average MAE between 0.2 and 0.3 can be considered as a small deviation. Further, we point out that we can see the tendency that our predicted spectrograms are smoother than the clean spectrograms themselves. As shown in Figure 3, we observe the presence of artifacts in the clean spectrogram that are missing in our predicted spectrograms. These artifacts represent changes in the air pressure in time intervals of 20 ms. We speculate that

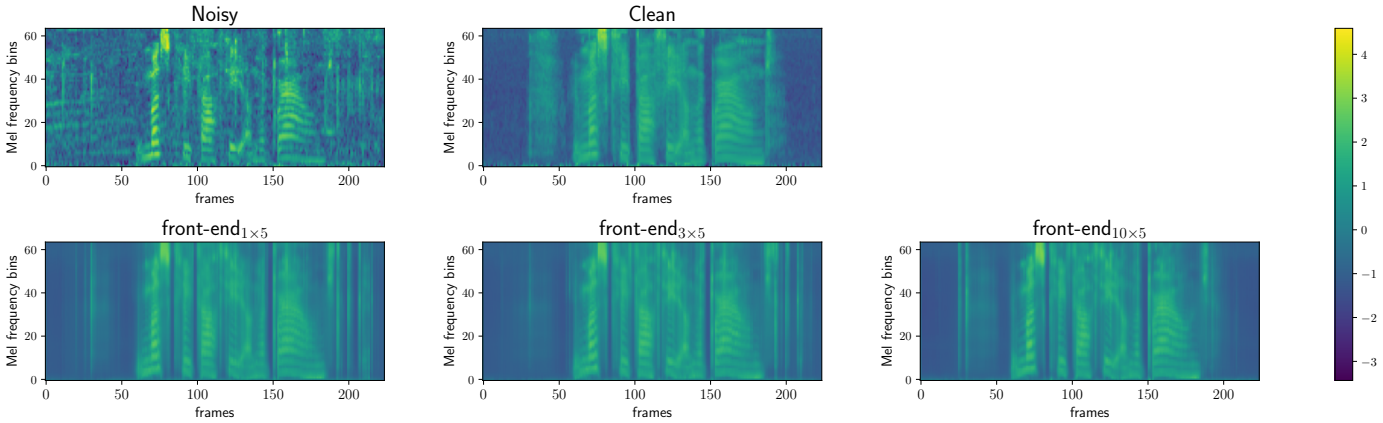

Fig. 3. Both upper spectrograms represent the normalized log Mel filterbanks of the noisy and clean signal. Our front-ends receive the noisy signal and are tasked to predict cleaner representations that are shown in the lower row.

these changes do not result from speech production. They could result from other sources like wind or present noise inherent in the system itself. Thus, they influence the MAE as our metric but do not necessarily influence the performance as a front-end.

We further investigate whether we can identify areas in our predictions that show a higher deviation than other areas. These deviations do also influence our MAE but could be irrelevant for speech recognition systems. Therefore, we concentrate on different Mel frequency bins. We calculate the MAE between clean and predicted spectrogram bin-wise for every sample. We observed a tendency that some Mel frequency bins are less reconstructed based on the speaker in all our front-ends through all samples. We observe that the female speaker has a higher MAE in her lower frequency bin than the male. This is the case for every single predicted spectrogram for every single front-end.

We speculate if our front-ends are less invested in reconstructing the lower bin because of the fundamental frequency. The lower Mel frequency bin represents the frequency domain between 0-57 Hz, and the second bin represents the frequency domain between 28-119 Hz. The fundamental frequency of a male voice is on average between 85-180 Hz, while the female voice is in a range between 165-250 Hz [23]. Further, only frequencies above the fundamental frequency hold information concerning speech recognition. It can be argued that the lower bin does not contribute to speech recognition. An ASR system could interpret these bins as "noise" for its respective task and learn to ignore them. Such behavior would explain why our front-ends were not able to reconstruct these bins from the ASR's activations.

Afterwards, we manually investigated sample outliers that show a higher MAE than other samples. We notice that in many of these samples inhalings are not reconstructed in our predicted spectrograms. We assume that these inhalings could be the source for the high MAE. On the other hand, inhaling is not irrelevant for speech recognition. Therefore, our front-ends could have adopted this behavior from Jasper.

TABLE II
THIS TABLE SHOWS THE RESULTS CONCERNING THE WER(%) OF JASPER. WE TESTED ON THE NSD TEST SET AND PROVIDED JASPER EITHER WITH THE NOISY SPEECH OR THE OUTPUT OF OUR FRONT-ENDS. WE CALCULATE THE WER CONCERNING THE SPEAKER, SNR, AND LOCATION.

		front-end		
	noisy	1×5	3×5	10×5
speaker				
male	15.7%	15.07%	14.95%	**14.18%**
female	20.88%	22.05%	22.22%	**19.72%**
SNR				
2.5	29.51%	27.41%	27.51%	**25.16%**
7.5	18.37%	18.71%	19.52%	**16.94%**
12.5	11.90%	12.57%	11.57%	**10.82%**
17.5	**13.78%**	16.22%	16.34%	15.33%
background noise (location)				
bus	14.02%	15.70%	15.54%	**13.14%**
cafe	29.99%	27.57%	27.80%	**26.13%**
living	19.58%	21.44%	21.61%	**17.83%**
office	**11.28%**	13.84%	13.66%	13.00%
psquare	17.24%	16.09%	16.21%	**15.36%**

3) WER at Use as Front-End: We test next how well our front-ends can be used for the pre-trained Jasper DR 10×5 model as back-end. We use again the pre-trained Jasper DR 10×5 model that we use in the encoder of our front-ends. Again, we test considering different speakers, SNRs, and locations. The results can be seen in Table II.

We observe that our front-end$_{10\times5}$ outperforms the other front-ends. Further, we notice that the performance of the front-end$_{1\times5}$ and front-end$_{3\times5}$ differ concerning the condition. Our front-end$_{1\times5}$ outperforms our front-end$_{3\times5}$ in an SNR of 2.5 and 7.5 dB. On the other hand, the front-end$_{3\times5}$ shows a lower WER on the samples with an SNR of 12.5 dB.

We notice two conditions where our front-end$_{10\times5}$ decreases the performance of the ASR system. We see a decrease in the WER for the SNR 17.5 and the background location "office". We started to investigate "office" through listening.

We evaluate the background noise "office" as less prevalent than other types of background noise in the audio samples. We speculate that our front-end decreases the performance of an ASR if there is little or no noise.

B. Experiment II

We hypothesize that if we have externalized Jasper's capability of removing noise, a smaller ASR system does not have to learn it. Effectively, it could concentrate on learning patterns in speech data. Further, we want to address the question of whether we have isolated Jasper noise-robustness. An ASR system that uses our front-end should still perform better in noisy environments, even if it was trained on clean data.

1) Dataset, Model, and Loss: As for the choice of the ASR systems, we use the Jasper 10×3 architecture. For a first experiment, we avoid changing the architecture because they could require different processing mechanisms concerning noise. The 10×3 model is suggested by NVIDIA to work on a mini-processor called "Jetson Nano". Thus, we are provided with an architecture that was already tested. Jasper 10×3 uses about 20 convolutional layers less than the 10×5 architecture and showed to be less performant than the DR 10×5 architecture.

As for the training set, we choose the LibriSpeech training set. This dataset provides clean speech data and transcriptions of 80 speakers. Further, it roughly provides an equal amount of male and female speakers. Each speaker has about 25 to 30 minutes which results in approximately 1000 hours of speech samples. Further, we use the clean development set for validation. This subset contains 5.4 hours of speech data and contains an equal amount of speakers.

For testing, we choose two datasets. On the one hand, we aim to investigate whether our front-end supports another Jasper model in noisy environments after training. For this purpose, we use the test set of the NSD dataset. On the other hand, we investigate how our trained Jasper model performs with clean data. Therefore, we use the test-clean and test-other subset of LibriSpeech. Roughly, both subsets contain 5 hours of speech and have the same amount of female and male speakers.

2) Results: We train two Jasper 10×3 models on LibriSpeech. Our baseline ASR system uses the officially recommended hyper-parameters for training [7]. These recommendations include a data augmentation technique that randomly masks parts of the spectrograms. It is shown that this data augmentation technique improved the performance of the best Jasper model [7]. Further, it could be argued that the masking can be considered as adding noise. Works indicate that training with a data augmentation technique leads to more noise robustness [5], [24]. Our second model also trains on the recommended values provided by NVIDIA but uses our front-end$_{10\times5}$ instead of the recommended data augmentation technique. Both models train over 50 epochs with CTC loss. We train both with a batch size of 24 on 8 GPUs.

We see the progress of training in Figure 4 over time. We observe that the ASR system that uses our front-end has a less stable learning curve than the baseline model. However, it outperforms the baseline ASR system in almost every step on the validation set. The less stable training curve could be explained by our front-end removing Mel frequency bins concerning the speaker. The ASR system that trains with our front-end receives more varying values in these bins. That could animate the network to learn speech-related patterns in earlier layers.

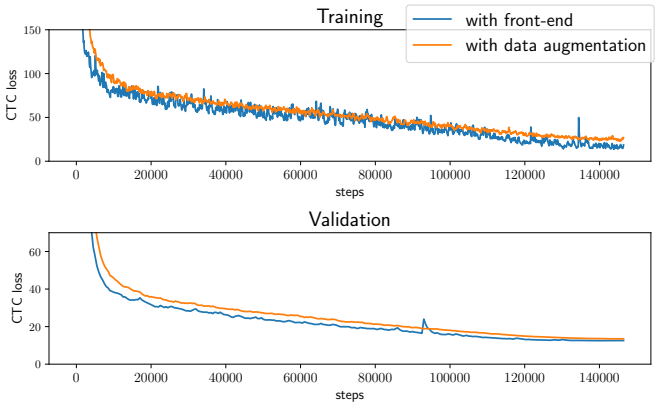

Fig. 4. We see the training curve of our front-end (upper) and the validation curve (lower) with CTC loss on LibriSpeech.

TABLE III
THIS TABLE SHOWS THE WER(%) OF BOTH JASPER 10×3 SYSTEMS ON THE NSD TEST SET. ONE SYSTEM USED OUR FRONT-END FOR TRAINING AS WELL AS TESTING, THE BASELINE SYSTEM TRAINED ON THE RECOMMENDATIONS OF NVIDIA.

	with front-end	baseline
NSD		
speaker		
male	**29.82%**	38.02%
female	**43.01%**	48.64%
SNR		
2.5	**47.53%**	57.18%
7.5	**38.21%**	44.47%
12.5	**27.94%**	35.05%
17.5	**32.96%**	38.86%
background noise (location)		
bus	**31.87%**	37.13%
cafe	**45.34%**	55.41%
living	**38.29%**	45.88%
office	**32.10%**	36.42%
psquare	**37.67%**	44.38%
LibriSpeech		
test-clean	**10.91%**	14.23%
test-other	**15.56%**	33.65%

We see in Table III that the model that trained with our front-end shows a better performance in every category of the NSD. We further note that it also shows superior performance on clean data. The difference in performance is especially noticeable on the LibriSpeech test-other dataset that is considered more challenging.

V. Conclusion

Our results indicate that we have leveraged the noise processing capabilities of a pre-trained ASR neural network for sustainable use in a new front-end. Our front-ends remove background noise and show the tendencies to remove speech-irrelevant features.

Further, we point out that our experiments show very different results than the experiment of *Li et al.* [5]. While they observed the reconstruction of less speaker information with more reused layers, our deepest front-end, the front-end$_{10\times5}$, outperformed the others in every tested aspect. We argue that this results from our training scheme combined with Jasper. Our front-ends have probably learned which kernels are productive for filtering and avoided counterproductive ones through Jasper's skip connections.

Furthermore, we demonstrate the benefits of our approach to modularizing Jasper into a front-end. We show that the front-end supports a pre-trained Jasper DR 10×5 model in noisy environments. Additionally, we demonstrate that our front-end supports another ASR system if it uses the front-end for training. We observe an increased performance for noisy and clean speech. That raises hope that our front-end will have a similar effect on ASR systems that work on the same speech features as Jasper.

While we have shown that our front-end works well on unknown but predictable sources of noise, we expect that it will perform well on all kinds of additive noise because of its convolutional architecture. Our front-end reconstructs each frame in the output based on multiple frames in the input. As an example, our deepest front-end considers more than 51 input frames for reconstructing one frame. On the other hand, our front-end has to be tested on noise that is not additive like reverberation. We propose that for future work.

Our approach indicates that we can use the noise-filtering capabilities of a trained ASR system to create a new front-end that provides cleaner spectral representations. As a result, we have modularized an end-to-end model. We assume that our front-end can also contribute to tasks like emotion recognition, speaker identification, or speech detection if it was adapted. Furthermore, we hypothesize that our training scheme can be used for other systems in other domains. We could modularize several E2E systems if the architecture is beneficial and the data are given. We are interested to see which impact our work can have on training large neural networks in the future.

References

[1] H. Fletcher and W. A. Munson, "Loudness, its Definition, Measurement and Calculation," *The Bell System Technical Journal*, vol. 12, no. 4, pp. 377–430, Oct. 1933.

[2] E. Zwicker, "Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen)," *The Journal of the Acoustical Society of America*, vol. 33, no. 2, pp. 248–248, Feb. 1961.

[3] S. Siddiqui, G. Rasool, R. P. Ramachandran, and N. C. Bouaynaya, "Using Deep Speech Recognition to Evaluate Speech Enhancement Methods," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, pp. 1–7.

[4] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," *arXiv:1412.5567 [cs]*, Dec. 2014.

[5] C.-Y. Li, P.-C. Yuan, and H.-Y. Lee, "What does a network layer hear? Analyzing hidden representations of end-to-end ASR through speech synthesis," *arXiv:1911.01102 [cs, eess]*, Nov. 2019.

[6] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. South Brisbane, Queensland, Australia: IEEE, Apr. 2015, pp. 5206–5210.

[7] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, "Jasper: An End-to-End Convolutional Neural Acoustic Model," *arXiv:1904.03288 [cs, eess]*, Aug. 2019.

[8] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully Convolutional Speech Recognition," *arXiv:1812.06864 [cs]*, Apr. 2019.

[9] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2Letter: An End-to-End ConvNet-based Speech Recognition System," *arXiv:1609.03193 [cs]*, Sep. 2016.

[10] C. Valentini-Botinhao, "Noisy speech database for training speech enhancement algorithms and TTS models," *http://parole.loria.fr/DEMAND/*, Aug. 2017.

[11] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, "Investigating RNN-based speech enhancement methods for noise-robust Text-to-Speech," in *9th ISCA Speech Synthesis Workshop*, Sep. 2016, pp. 146–152.

[12] ——, "Speech Enhancement for a Noise-Robust Text-to-Speech Synthesis System Using Deep Recurrent Neural Networks," in *Interspeech 2016*, Sep. 2016, pp. 352–356.

[13] C. Veaux, J. Yamagishi, and S. King, "The voice bank corpus: Design, collection and data analysis of a large regional accent speech database," in *2013 International Conference Oriental COCOSDA Held Jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE)*. Gurgaon, India: IEEE, Nov. 2013, pp. 1–4.

[14] J. Thiemann, N. Ito, and E. Vincent, "The Diverse Environments Multi-channel Acoustic Noise Database (DEMAND): A database of multichannel environmental noise recordings," in *ICA 2013 Montreal*, Montreal, Canada, 2013, pp. 035 081–035 081.

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Mar. 2015.

[16] V. Liptchinsky, G. Synnaeve, and R. Collobert, "Letter-Based Speech Recognition with Gated ConvNets," *arXiv:1712.09444 [cs]*, Feb. 2019.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2016.

[18] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, H. Nguyen, Y. Zhang, and J. M. Cohen, "Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks," *arXiv:1905.11286 [cs, stat]*, Feb. 2020.

[19] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway Networks," *arXiv:1505.00387 [cs]*, Nov. 2015.

[20] C. Donahue, B. Li, and R. Prabhavalkar, "Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 5024–5028.

[21] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-Aware Neural Language Models," in *2016 AAAI Conference on Artificial Intelligence*, vol. 30, 2016, p. 9.

[22] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," *arXiv:1703.10135 [cs]*, Apr. 2017.

[23] I. R. Titze, *Principles of Voice Production*. Englewood Cliffs, N.J: Prentice Hall, 1994.

[24] L. Tóth, G. Kovács, and D. Van Compernolle, "A perceptually inspired data augmentation method for noise robust cnn acoustic models," in *International Conference on Speech and Computer*. Springer, 2018, pp. 697–706.