DISS. ETH NO. 26351

PARAMETER UNCERTAINTY AND MULTI-SENSOR ATTENTION MODELS FOR END-TO-END SPEECH RECOGNITION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

STEFAN BRAUN

M.Sc., Karlsruhe Institute of Technology

born on 10.03.1989

citizen of Germany

accepted on the recommendation of

Prof. Shih-Chii Liu
Prof. Richard Hahnloser
Prof. Nima Mesgarani

2019

ABSTRACT

Over the past decades, the dominant approach towards building automatic speech recognition (ASR) systems has been a complex combination of separately optimized pre-processing, acoustic model and language model components. The recently proposed end-to-end models for ASR present a significant simplification over conventional ASR systems. End-to-end models transcribe input speech to output text with a single neural network that is optimized in a single training stage. While the single model and training stage are a welcome simplification of the ASR system, they are also mostly incompatible with past research that went into optimizing the separate components of conventional ASR systems. Furthermore, the monolithic neural network structure in end-to-end models remains a black box with millions of parameters, and the contribution of specific parameters to the model accuracy is hardly understood. In consequence, the accuracy of conventional ASR systems is still higher, and end-to-end models require new strategies to improve.

This thesis has the objective to advance the state-of-the art in end-to-end models for ASR, with a focus on improving noise robustness and model interpretability. The contributions cover novel training strategies and neural network architectures, and three main contributions can be identified. First, a curriculum learning strategy is presented that improves the noise robustness over conventional training methods. The network training follows a signal-to-noise ratio (SNR) curriculum that starts training at low SNR levels and gradually exposes the network to higher SNR levels as training proceeds. Second, a sensory attention mechanism is integrated into the end-to-end model, adding only a fraction of the total parameters to the model. The attention mechanism allows the model to extract information from multiple input sensors and dynamically tune its attention towards less noisy sensors for improved accuracy. The attentional signal is highly interpretable as it correlates with the sensor noise level. Third, the entire model architecture is changed, replacing the deterministic neural network parameters with probabilistic parameters. All network parameters are sampled from probability distributions with a learned degree of uncertainty, and the uncertainty information is interpreted as a proxy measure for parameter importance. The parameter importance information is used in parameter pruning for saved computation and domain adaptation for increased noise robustness.

ZUSAMMENFASSUNG

Im Laufe der letzten Jahrzehnte bestanden Systeme für die automatische Spracherkennung aus einer komplexen Kombination von separat optimierten Komponenten für die Datenvorverarbeitung, akustische Modellierung und sprachliche Modellierung. In jüngerer Zeit wurden Spracherkennungssysteme vorgestellt die auf dem „end-to-end" Prinzip basieren und eine deutliche Vereinfachung gegenüber konventionellen Spracherkennungssystemen darstellen. In „end-to-end" Modellen geschieht die Umwandlung von Sprache zu Text mithilfe eines einzigen künstlichen neuronalen Netzwerkes, welches in einer einzigen Lernphase optimiert wird. Sowohl Modell als auch Lernphase vereinfachen das Spracherkennungssystem, sind allerdings auch inkompatibel mit Jahren an Forschung, welche die separaten Komponenten in konventionellen Spracherkennungssystemen optimiert hat. Zusätzlich ist das neuronale Netzwerk eine Art „Blackbox" mit Millionen an Parametern, und die Bedeutung einzelner Parameter für die Spracherkennungsfähigkeit des Modells ist kaum verstanden. In der Konsequenz bleiben konventionelle Spracherkennungssysteme bei der Genauigkeit überlegen, und „end-to-end" Modelle benötigen neue Strategien zur Verbesserung.

Diese Arbeit legt den Fokus auf die Verbesserung von „end-to-end" Modellen für die Spracherkennung, insbesondere was die Robustheit in verrauschten Umgebungen und was die Interpretierbarkeit des Modells betrifft. Die wissenschaftlichen Beiträge umfassen neue Strategien für neuronale Netzwerke in den Gebieten Lernphase und Architektur, und drei Haupt-Beiträge sind zu nennen. Der erste Beitrag präsentiert eine Curriculum-basierte Strategie für das Lernen von neuronalen Netzwerken, welche die Robustheit in verrauschten Umgebungen gegenüber konventionellen Lernmethoden verbessert. Die Lernphase des Netzwerks folgt einem Curriculum welches die Lernphase mit besonders rauschbehafteten Sprachdaten beginnt, und im weiteren Verlaufe der Lernphase das Signal-zu-Rausch Verhältnis der Sprachdaten zu höheren Werten hin erweitert. Der zweite Beitrag behandelt einen Mechanismus für sensorische Aufmerksamkeit („sensory attention mechanism") welcher sich zur Integration in das „end-to-end" Modell eignet und nur einen Bruchteil der ursprünglichen Anzahl der Parameter zum Modell hinzufügt. Der „attention mechanism" erlaubt es dem Modell von mehreren Sensoren gleichzeitig Informationen zu extrahieren, und in dynamischer Weise die Aufmerksamkeit auf weniger rauschbehaftete Sensoren zu lenken um eine erhöhte Genauig-

keit zu erzielen. Das Aufmerksamkeits-Signal ist intuitiv interpretierbar und korreliert mit dem Niveau des Rauschens, welchem ein Sensor ausgesetzt ist. Der dritte Beitrag ändert die gesamte Architektur des Modelles und ersetzt deterministische Parameter des neuronalen Netzwerkes mit probabilistischen Parametern. Alle Netzerkparameter werden aus Wahrscheinlichkeitsverteilungen gezogen die einen erlerntes Ausmaß an Zufälligkeit aufweisen. Das Ausmaß der Zufälligkeit wird als stellvertretende Größe für die Wichtigkeit von Parametern betrachtet. Diese Information der Wichtigkeit von Parametern wird für zwei Experimente benutzt. Zum einen für das Ersetzen von Parametern mit Nullwerten („parameter pruning“), wenn Rechenoperationen eingespart werden sollen. Zum anderen für die Umgebungs-Anpassung akustischer Modelle („domain adaptation“), wenn die Robustheit gegenüber Rauschen verbessert werden soll.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Shih-Chii Liu for her guidance, support and encouragement throughout the PhD studies. I would also like to thank my co-authors Daniel Neil, Enea Ceolini, Jithendar Anumula, Chang Gao, Ilya Kiselev and Tobi Delbruck for their strong commitment and effective collaboration. Further thanks go to my colleague and desk-neighbor Adrian Huber for many critical discussions over the past years. This work has received valuable feedback from various places and persons, notably the audio group, the sensors group, the Neuromorphic Processor Project team, and all the people from the Institute of Neuroinformatics at the University of Zürich and ETH Zürich. A special thanks goes to Richard Hahnloser and Nima Mesgarani for their support as co-examiners in the doctoral committee. This work would not have been possible without the generous funding from my sponsor Samsung, and computational resources provided by ETH Zürich and the University of Zürich. Finally, I would also like to thank my wife, family and friends for their support and encouragement over the past years.

CONTENTS

ACRONYMS

ACCAN	Accordion annealing
ASR	Automatic speech recognition
ATTACC	Attention accuracy
ATTCORR	Attention correlation
AVSR	Audio-visual speech recognition
CER	Character error rate
CFE	Convolutional front-end
CNN	Convolutional neural network
CTC	Connectionist temporal classification
DNN	Deep neural network
DTW	Dynamic time warping
GMM	Gaussian mixture model
HMM	Hidden Markov model
LSTM	Long short-term memory
LVCSR	Large-vocabulary continuous speech recognition
MVDR	Minimum variance distortionless response
PEM	Per-epoch noise mixing
RNN	Recurrent neural network
RNN-T	Recurrent neural network transducer
SNR	Signal-to-noise ratio
STAN	Sensor transformation attention network
STFT	Short-time Fourier transform
TER	Token error rate

WER Word error rate

WFST Weighted finite state transducer

WSJ Wall Street Journal

1

INTRODUCTION

This introductory chapter is split into two parts. The first Section 1.1 introduces the basic ASR task and the historical development of ASR systems towards end-to-end speech recognition. The following Section 1.2 discusses the background and novelty of the three main contributions of the thesis.

1.1 AUTOMATIC SPEECH RECOGNITION

The research field of automatic speech recognition (ASR) is clearly defined by its main task: to map input speech to an output text transcription. The standard ASR procedure for an example input-output pair is shown in Figure 1.1. The speech input consists of audio samples recorded by a microphone, and the text output consists of text tokens, e.g. characters or words. Both input and output are sequences of variable length. Given typical audio sampling rates, e.g. 16kHz, it is evident that the input and output sequences also have different lengths. Any ASR system will therefore have to map a longer sequence of inputs, to a shorter sequence of outputs. The mapping of a speech signal to a single word is referred to as *isolated word recognition*. In this classification task, each sample of the audio signal is considered to contribute to a single word label. The mapping of a speech signal to a sequence of text tokens is referred to as *continuous speech recognition*. Typically, there is no precise *alignment* available that would assign single text labels to segments of audio samples. In addition to classification, the ASR system also has to account for the speech-to-text alignment, which makes continuous speech recognition the more difficult task. From a broader perspective, continuous speech recognition can be interpreted as a sequence-to-sequence task, and is therefore related to other sequence-to-sequence tasks such as machine translation (text-to-text) [1] or speech synthesis (text-to-speech) [2].

Over the past decades, ASR research has steadily improved the recognition accuracy under growing task complexity. Three main developments can be identified: (1) a shift from template-matching to statistical modeling methods, (2) a substitution of the simpler isolated word recognition task with the continuous speech recognition task and (3) an increase in vocabulary sizes

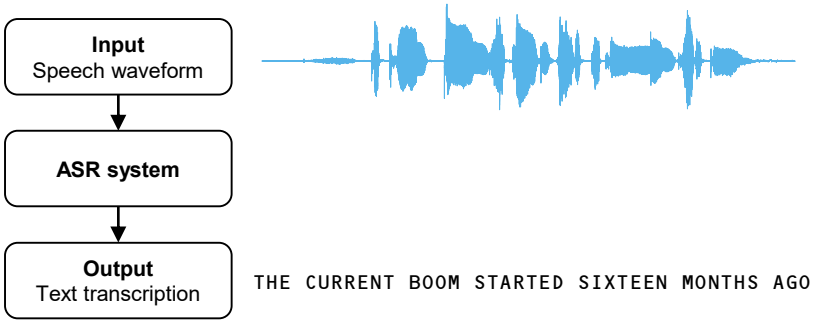


FIGURE 1.1: The standard ASR procedure. The ASR system maps an input speech waveform to an output text transcription. Note the difference in sequence length. In this example, the speech waveform consists of 66146 audio samples, sampled at a 16kHz sample rate. The text output consists of 37 characters, or 7 words.

from 10 words to virtually unlimited. The following paragraph presents a few important milestones in ASR history, and the interested reader is referred to [3] and [4] for more complete reviews.

In 1952, researchers at the Bell laboratories built the *Audrey* ASR system that performed spoken digit recognition with an algorithm that analyzed formant trajectories [5]. The vocabulary consisted of 10 units that represented the digits zero to nine. The Audrey system was limited to isolated digit recognition. For sequences of multiple digits, the speech recordings were split into isolated digit utterances by compulsory speaking pauses. Over the next years, isolated word recognition represented the main task in ASR research, and template-matching became a popular approach for classification. However, the temporal variability of word utterances presented a challenge for comparing word templates. In 1968, template-matching was significantly improved when the dynamic time warping (DTW) algorithm was first proposed for ASR [6]. The DTW algorithm aligns speech utterances with different durations to facilitate the comparison step in template-matching. By 1970, a DTW-based system was introduced for a 200 words vocabulary [7]. A major leap forward was established in 1971 when DARPA started the five year program *Speech Understanding Research* for continuous speech recognition and a 1000 words vocabulary. By 1976, *Harpy* from CMU [8] was the only system that met the DARPA performance requirements [3]. The Harpy system combined template-matching with finite state machines and used graph search techniques to find

the most likely text transcription. At the same time, Hidden Markov models (HMMs) were explored as an alternative statistical approach to ASR [9, 10], and also combined with language models to improve the recognition accuracy [11]. The HMMs provided a statistical modeling framework for continuous speech recognition, and have become the dominant approach to ASR since the 1980s. By 1989, IBM built an HMM-based continuous speech recognition system with a 5000 words vocabulary [12]. From thereon, the vocabulary size grew to virtually unlimited size, e.g. the *Switchboard* dataset from 1993 has a vocabulary of 45,000 words and is still used today.

Over the past decades, HMM-based ASR systems have received a steady stream of improvements. One improvement that is particularly interesting from a neural network perspective is the emerging of deep neural network (DNN)-HMM hybrids in 1990 [13]. Even though neural networks had been applied to ASR before, they were mostly restricted to isolated word recognition [14]. For continuous speech recognition, a major challenge remained unsolved: to derive an objective that allows neural network training when the temporal alignment between speech signal and text labels is not available. As a workaround, the hybrid approach combines DNNs for acoustic modeling with the HMM framework. The speech signal is split into frame segments of equal length, and for each frame, the DNN outputs a probability distribution over HMM states. The HMM states correspond to fragments of phonetic output units, and the output distribution is decoded in the HMM framework to generate text output. The HMM decoding deals with the temporal alignment and integrates external information from language models. Despite their early proposal, the real breakthrough for hybrids came only in 2012, when computational resources and training strategies had improved enough to train competitive DNNs [15]. This new generation of DNN-HMM hybrids achieved lower error rates than previous HMM systems that used Gaussian mixture models (GMMs) for acoustic modeling. Nowadays, DNN-HMM hybrids still represent the state of the art in ASR [16]. However, DNN-HMM hybrids are complex systems: they consist of multiple components which are optimized in different training stages [17]. For example, the DNN training alone is already a multi-stage process. To train the DNN on speech frames, each frame requires a target label - the HMM state. The labels are generated by a process called *forced alignment*. This alignment process requires a GMM-HMM ASR system that is trained in an iterative fashion until the alignment quality is empirically considered sufficient [15–17].

In recent years, end-to-end models have been proposed [18–27] that present a significant simplification over DNN-HMM hybrids in both model archi-

tecture and training process. End-to-end models transcribe input speech to output text within a single neural network that is optimized in a single training stage. The training process does not require an external alignment from speech frames to output labels. Instead, two approaches can be identified to deal with the alignment problem: (1) novel training objectives or (2) attention-based neural network architectures. Objective functions such as Connectionist temporal classification (CTC) [28] or recurrent neural network transducer (RNN-T) [29] consider all possible alignments with efficient dynamic programming approaches. Encoder-decoder models use an attention mechanism that learns an alignment function automatically during training [30]. The choice of the end-to-end approach has theoretical and practical implications on the ASR system, e.g. the suitability for language modelling or online speech recognition [21]. At this point of the introduction, the reader is referred to Chapter 2 for an extensive description of the CTC approach and a comparison to the RNN-T and encoder-decoder models.

While the relative simplicity of end-to-end models sounds very compelling, there is a significant cost attached: as of 2019, conventional DNN-HMM hybrids still achieve superior accuracy and remain the state of the art method in ASR [16]. A large part of the performance gap may be explained by the strict end-to-end criteria of a single training stage and network model. These criteria are very restrictive and make end-to-end models incompatible with decades of research that went into optimizing ASR systems based on DNN-HMM hybrids. Consequently, end-to-end models require new methods to improve, and this thesis proposes three contributions that are described in Section 1.2.

1.2 CONTRIBUTIONS

This thesis advances the state of the art in end-to-end models for ASR. The overall goals are improving noise robustness and model interpretability while respecting the end-to-end criteria of a single training stage and a single neural network model. This thesis provides three contributions that are explained in the following three sections.

1.2.1 *Curriculum Learning for Improved Noise Robustness*

The performance of ASR systems has improved significantly over the past decades, reaching human-level performance in clean conditions without background noise [20]. However, the performance in noisy environments still leaves room for improvement [20, 31]. Over the past decades, a multitude of methods has been proposed to improve the noise robustness of ASR systems [32]. Some methods that have been successfully used with conventional DNN-HMM hybrids include multi-channel enhancement techniques [33, 34], denoising methods [35], source separation methods [36, 37], feature transformation methods such as fMLLR [38] or auditory level features [39]. However, most of these strategies add components to the ASR system that require disjoint optimization stages. While disjoint optimization is typical for conventional DNN-HMM hybrids, it is not compatible with the end-to-end ASR paradigm.

For end-to-end models, the two main options to improve noise robustness are (1) novel neural network architectures and (2) novel training methods. Recent research has shown improved noise robustness when formerly hand-tuned feature-processing stages are replaced with learnable neural network layers. Typical examples include feature extraction at the raw waveform level via convolutional neural networks (CNNs) [40–43] or convolutional front-ends (CFEs) for noise robust feature processing [20, 44]. When suitable, entire pre-processing algorithms such as multi-channel enhancement with beamforming may be modelled with neural networks and integrated into the end-to-end model [45]. As an alternative to architectural changes, training on noisy data is an established method of increasing the noise robustness. In related work on DNN training methods, noisy training sets with a range of SNR values e.g. 10 dB - 20 dB [46] or 0 dB - 30 dB [40] are used during training. This training strategy is referred to as *multi-condition* training. Other training methods such as dropout [47] - originally intended to improve regularisation in neural networks - have been shown to also improve noise robustness for ASR applications.

This thesis chapter considers a single input channel and introduces a novel *curriculum learning* strategy to improve the accuracy of end-to-end models in noisy conditions. While curriculum learning has been proposed before to increase the accuracy of neural networks [48], this is the first contribution that leverages curriculum learning for noise robust ASR. The proposed curriculum learning method follows a SNR curriculum that starts network training at 0 dB SNR, and then gradually expands the range towards 50 dB SNR as training continues. This SNR range is significantly larger than the range in previous works. The curriculum learning strategy results in models with improved accuracy in noisy conditions when compared to conventional multi-condition training methods. All experiments and results are described in Chapter 3.

1.2.2 *Attention-driven Multi-Sensor Selection*

Most ASR research with end-to-end models has focused on single-channel input only. The single-channel setup is the most simple input configuration, as it consists of a single microphone sensor that records a single audio modality. However, real-world ASR systems (Amazon Echo, voice control systems in cars etc.) leverage multiple sensors to deal with speech in noisy conditions. The additional hardware cost from multiple sensors is offset by two advantages. First, multi-sensor setups can be exploited by multi-channel processing or sensor fusion strategies to increase the recognition accuracy. Second, multiple sensors are inherently redundant, and increase the robustness and fault tolerance of the ASR system. Motivated by these useful advantages, this thesis chapter is therefore dedicated to combine multi-sensor input with end-to-end models.

When considering only the audio modality, multi-channel processing generally integrates multiple input channels into a single, enhanced channel that provides a cleaner signal for classification. In this context, conventional beamforming algorithms are widely used for multi-channel setups [34], but they introduce a separate beamforming processing stage which is typically not compatible with the end-to-end paradigm. Alternate approaches for multi-channel integration are based on methods that leverage CNNs for channel combination [49–51], that learn a beamforming function with neural networks [52–57], and attention mechanisms that focus on higher SNR channels [58]. While these methods are differentiable and suitable for joint optimization in an end-to-end model, they were usually combined with conventional hybrid ASR approaches. Two recent studies [45, 59] have examined multi-channel ASR

and meet the criteria of end-to-end models. In both studies, inputs from the multiple channels are combined into a single representation that is used for the ASR task. In one case, a *neural beamformer* is used to combine the channels [45] and in the second case, a *sensory attention mechanism* [59] is used instead. While both approaches show promising performance compared to conventional beamforming, the neural beamformer shows benefits such as invariance to channel re-ordering and robustness to channel configurations.

When increasing the scope to multi-modal sensor setups, then the audio-visual input configuration is the most explored one, and the ASR task is replaced with the audio-visual speech recognition (AVSR) task. The additional video modality is especially useful in noisy acoustic conditions, where a video recording of the speakers mouth region can be used for lip reading to provide a high transcription quality. In order to integrate information from both modalities, an AVSR system has to combine the information from audio and video streams. This process is generally referred to as *sensor fusion*. Two recent studies have covered end-to-end models for AVSR [60, 61], and both combined the audio and video streams with a feature concatenation layer in the neural network. The concatenation layer performs sensor fusion by stacking features together, and leaves the network to learn its preferred feature dimensions. The concatenation operation is also used in other audio-related tasks with audio-visual input, e.g. speech enhancement [62] and source separation [63]. Other network operations that may be used for sensor fusion are convolutions or averaging, and have been explored in a video classification scenario [64].

From the perspective of end-to-end models, the integration of multiple sensors of arbitrary modalities is feasible as long as the fusion operation is differentiable. This condition is met for standard fusion operations such as concatenation or convolution. However, these fusion operations are *static*: once the network weights are learned towards each sensor, they are not changed any more, and a network may develop preferences towards a particular sensor. In real world scenarios, where the signal recorded by the preferred sensor may be temporarily more noisy than expected, an adaptive strategy is advantageous. One such strategy is the *sensory attention mechanism*, which can be considered as a *dynamic* fusion operation. The sensory attention mechanism first weighs and then sums multiple input sensors into a single representation. The sensor-specific attention weights are computed by neural networks that are integrated into the end-to-end model such that a single training process is sufficient. This attention mechanism allows the model to dynamically tune its attention towards less noisy sensors for improved

accuracy. The attention weights are highly interpretable, as their magnitude indicates how much the model is focusing on a specific sensor. The dynamic tuning and understandable attention weights are both properties that are well aligned with the thesis goals of improving noise robustness and model interpretability.

This thesis chapter explores a *sensory attention mechanism* to process multi-sensor input within end-to-end models. The sensory attention mechanism follows an improved design strategy compared to [59], with three main differences. First, it is defined in a modality-independent way, extending the scope from multi-channel ASR to multi-modal AVSR. Second, it uses default neural network units such as long short-term memorys (LSTMs) instead of a custom-designed neural network cell. Third, it supports a new operation mode that is invariant to the re-ordering of the audio channels in a multi-channel ASR setup. The proposed attention mechanism is embedded into a sensor transformation attention network (STAN). STANs provide a novel end-to-end framework that supports multi-sensor input of arbitrary modalities.

In terms of evaluation, this study covers both multi-channel ASR with real-world noise and multi-modal AVSR with synthetic noise. The multi-modal experiment presents two novelties, as this is the first study that evaluates sensory attention for AVSR or for a synthetic noise environment. The synthetic noise allows to establish a ground truth for the sensor noise level and to measure the correlation between noise level and attentional signal.

The experimental sections compare the sensory attention mechanism against end-to-end strategies for multi-sensor processing, notably sensor concatenation and sensor averaging, and also conventional beamforming algorithms. Across all experiments, the sensory attention mechanism performs on par or better than concatenation or averaging, with the additional benefit of providing a highly interpretable attentional signal. Compared to conventional beamforming algorithms, the sensory attention mechanism achieves higher error rates, but remains compatible with the end-to-end paradigm and significantly reduces the model complexity. The complete set of experiments and results is described in Chapter 4.

1.2.3 *Parameter Uncertainty for End-to-end Models*

End-to-end models for ASR are *parametric* models: they provide tunable parameters for optimization, which correspond to the weights and biases of neural network units. Conventional end-to-end models use *deterministic*

parameters, i.e. each parameter is a real value. Recent end-to-end models are composed of 10M (EESEN [22]) over 30M (Wav2Letter [43]) to 100M (DeepSpeech2 [20]) parameters.

While deterministic parameters encode the parameter magnitude, there is no direct encoding of the parameter uncertainty or the parameter importance to solve the task it was trained on. However, parameter importance is valuable information for neural networks. For parameter pruning, the importance information could help to select less important parameters for pruning and minimize the impact of the pruning process on task performance. Continual learning scenarios like speaker or domain adaptation are further areas that benefit from the importance information[1]. Domain adaptation is a process that improves the performance of ASR systems on unseen speech domains, e.g. when a network is trained on clean speech but is tested on noisy speech. Usually, the trained network is adapted on a small adaptation set, and the training set is not accessible during adaptation. During adaptation, the trained parameters get overwritten and the performance on the original training domain might be reduced. This phenomenon is referred to as *catastrophic forgetting* in literature [65]. By conditioning parameter updates on the importance information, the adaptation process could generate a single model that achieves high accuracy on the training and adaptation domains.

Recent work has explored parameter uncertainty in neural networks by encoding parameters in a *probabilistic* fashion [66–68]. Probabilistic neural networks sample parameters from probability distributions learned on training data, and each parameter exhibits a learned degree of uncertainty. The inherent parameter uncertainty makes probabilistic networks less sensitive to parameter perturbations and less prone to overfitting [66, 68]. The relation between the magnitude of a parameter and its uncertainty allows one to establish parameter-specific SNR levels. Previous studies show that there is a high correlation between parameter SNR and parameter importance as demonstrated in pruning experiments for tasks other than ASR [66, 67]. To the best of the authors knowledge, only one study investigated probabilistic neural networks for end-to-end ASR [68]. The probabilistic network was derived in a variational inference framework from a Bayesian perspective. The evaluation was carried out using a single probabilistic network on the 5h TIMIT dataset with a focus on parameter pruning.

This thesis chapter proposes an alternative derivation of probabilistic networks from a parameter perspective, without requiring a Bayesian inter-

1 The following description only mentions domain adaptation for the sake of readability, but could also be applied to speaker adaptation.

pretation of the model. The set of ASR tasks is extended to include domain adaptation from clean speech to noisy speech. In order to prevent forgetting of the original task, a novel SNR-based regularization scheme is proposed to condition parameter updates on parameter importance. Furthermore, this is the first study that evaluates probabilistic neural networks with distinct SNR levels. Networks with different SNR levels are studied with respect to how they tolerate pruning of parameters and how the level of catastrophic forgetting changes across these networks during domain adaptation. The complete study on parameter uncertainty is described in Chapter 5.

2

END-TO-END SPEECH RECOGNITION WITH CONNECTIONIST TEMPORAL CLASSIFICATION

This thesis evaluates all three contributions on end-to-end models that are based on the CTC approach. Therefore, this chapter is dedicated to presenting the CTC objective function [28, 69] in a more detailed fashion. Note that despite only being evaluated on CTC models, the thesis contributions are compatible with other models that full-fill the end-to-end criteria, such as RNN-T and encoder-decoder architectures. The interested reader is referred to [21] for a comparative study on all three models, and to Section 2.4 for a shorter discussion on the differences.

2.1 MODEL COMPONENTS

The ASR task is defined as the mapping of a speech signal to a text output. The speech input is assumed as a length-T sequence $\mathbf{x} = \{x_1, ..., x_T\}$ of audio samples $x_t \in \mathbb{R}$. The target output is assumed as a length-U sequence $\mathbf{y} = \{y_1, ..., y_U\}$ of text tokens $y_u \in \mathbf{v}$, and the vocabulary $\mathbf{v} = \{v_1, ..., v_N\}$ consists of $n = 1...N$ unique text tokens v_n [1].

A CTC model maps the speech input sequence $\mathbf{x}$ to a length-T^* output sequence $\mathbf{h} = \{h_1, ..., h_{T^*}\}$ of probability vectors $h_\tau \in \mathbb{R}^N$. Every vector h_τ consists of $n = 1...N$ entries, $h_\tau^{v_n}$, and each entry corresponds to the probability of the text token v_n at time step τ. The probabilities are expected to sum up to 1 at each time step τ, so $\sum_{n=1}^{N} h_\tau^{v_n} = 1$. The mapping function $h_\theta : \mathbf{x} \mapsto \mathbf{h}$ is usually modelled by neural networks that provide a set of tunable parameters θ. The output sequence length, T^*, can be shorter than the input sequence length, T, when typical ASR processing steps (splitting a waveform into frames, downsampling etc.) are used.

To summarize, $\mathbf{x}$ is the input sequence, h_θ is the neural network, $\mathbf{h}$ is the network output sequence and $\mathbf{y}$ is the target output sequence. Note that in literature, the mapping function h_θ is typically referred to as *encoder* or *acoustic model* [21].

1 For example, when using characters as vocabulary, then $v = \{A, ..., Z\}$ and $N = 26$.

2.2 TRAINING

During training, the CTC objective attempts to maximize the conditional probability $P(\mathbf{y}|\mathbf{h})$, i.e. the probability of the target sequence $\mathbf{y}$ given the network output sequence $\mathbf{h}$. The key challenge in computing this probability lies in both $\mathbf{h}$ and $\mathbf{y}$ having different sequence lengths of T^* and U, respectively. To match the length of $\mathbf{y}$ to $\mathbf{h}$, CTC introduces two measures. First, the blank token ϵ, which corresponds to an empty output, may be introduced at arbitrary steps of $\mathbf{y}$. Second, the tokens in $\mathbf{y}$ may be repeated arbitrarily before advancing to the next token. After sufficient insertion of blank tokens and repetitions, a length-T^* modification of $\mathbf{y}$ is referred to as an *alignment*. By definition, an alignment is a length-T^* sequence $\mathbf{a} = \{a_1, ..., a_{T^*}\}$ of text tokens $a_\tau \in \{v_1, ..., v_N, \epsilon\}$. An example is shown in Figure 2.1. A mapping function $\mathcal{A} : \mathbf{a} \mapsto \mathbf{y}$ is defined between $\mathbf{a}$ to $\mathbf{y}$, that first removes repeated tokens and then blank tokens in $\mathbf{a}$. The order is crucial, as a blank token between two identical tokens allows to model text with double tokens[2]. When $U < T^*$, there are many possible alignments $\mathbf{a}$ that correspond to the same desired target sequence $\mathbf{y}$ (see appendix A.1 for a discussion of the combinatorics). The set of all alignments that map to the desired target, $\{\mathbf{a} \in \mathcal{A}^{-1}(y)\}$, is referred to as the set of *valid* alignments. When computing the probability of the target sequence, CTC considers all valid alignments as defined in Eq. (2.1):

$$P(\mathbf{y}|\mathbf{h}) = \sum_{\mathbf{a} \in \mathcal{A}^{-1}(y)} P(\mathbf{a}|\mathbf{h}) \tag{2.1}$$

To compute the probability $P(\mathbf{a}|\mathbf{h})$ of a single alignment $\mathbf{a}$ given the network output $\mathbf{h}$, CTC assumes *conditional independence* between the network outputs over time, resulting in Eq. (2.2).

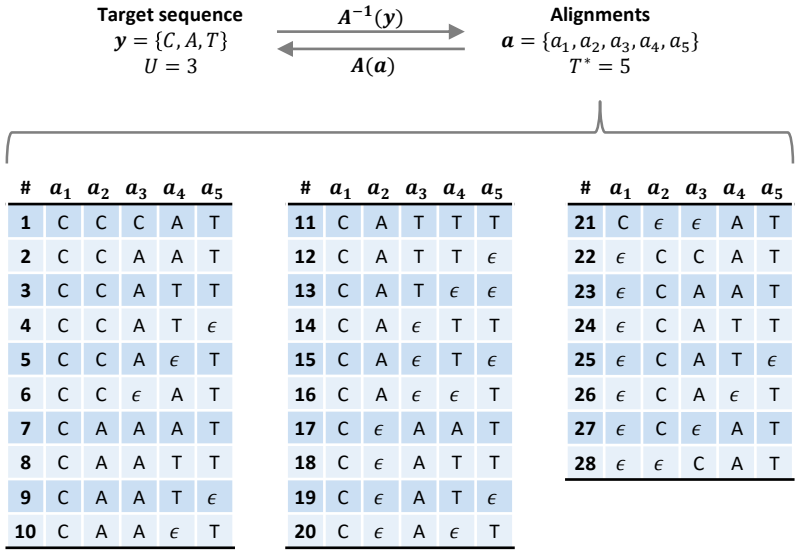
$$P(\mathbf{a}|\mathbf{h}) = \prod_{\tau=1}^{T^*} h_\tau^{a_\tau} \tag{2.2}$$

Plugging Eq. (2.2) in Eq. (2.1) yields the final CTC probability as defined in Eq. (2.3):

$$P(\mathbf{y}|\mathbf{h}) = \sum_{\mathbf{a} \in \mathcal{A}^{-1}(y)} \prod_{\tau=1}^{T^*} h_\tau^{a_\tau} \tag{2.3}$$

The summation over all paths in Eq. (2.3) can be computed by an efficient dynamic programming algorithm. To train the neural network, the conditional

2 e.g. `HELLO` or `GOOD`



#	a_1	a_2	a_3	a_4	a_5
1	C	C	C	A	T
2	C	C	A	A	T
3	C	C	A	T	T
4	C	C	A	T	ϵ
5	C	C	A	ϵ	T
6	C	C	ϵ	A	T
7	C	A	A	A	T
8	C	A	A	T	T
9	C	A	A	T	ϵ
10	C	A	A	ϵ	T

#	a_1	a_2	a_3	a_4	a_5
11	C	A	T	T	T
12	C	A	T	T	ϵ
13	C	A	T	ϵ	ϵ
14	C	A	ϵ	T	T
15	C	A	ϵ	T	ϵ
16	C	A	ϵ	ϵ	T
17	C	ϵ	A	A	T
18	C	ϵ	A	T	T
19	C	ϵ	A	T	ϵ
20	C	ϵ	A	ϵ	T

#	a_1	a_2	a_3	a_4	a_5
21	C	ϵ	ϵ	A	T
22	ϵ	C	C	A	T
23	ϵ	C	A	A	T
24	ϵ	C	A	T	T
25	ϵ	C	A	T	ϵ
26	ϵ	C	A	ϵ	T
27	ϵ	C	ϵ	A	T
28	ϵ	ϵ	C	A	T

FIGURE 2.1: The 28 valid alignments when mapping the length-3 target sequence $\mathbf{y} = \{C, A, T\}$ to the length-5 alignment $\mathbf{a} = \{a_1, a_2, a_3, a_4, a_5\}$.

probability in Eq. (2.3) provides a derivative with respect to the final network output layer, i.e. $\frac{\delta P(\mathbf{y}|\mathbf{h})}{\delta h_\tau^{v_n}}$. The dynamic programming algorithm and the computation of the derivatives are both discussed in [28].

2.3 INFERENCE

During inference, the network receives a speech input sequence $\mathbf{x}$ and outputs the token probability sequence $\mathbf{h}$. To generate a hypothetic text transcription $\tilde{\mathbf{y}}$ from $\mathbf{h}$, a decoding function $\mathcal{D} : \mathbf{h} \mapsto \tilde{\mathbf{y}}$ is used. In literature, various decoding algorithms have been proposed for CTC models, e.g. greedy decoding [28] or beam search [22]. The greedy decoding algorithm is used in most experiments of this thesis and therefore presented in the following paragraph.

At each time step τ, the greedy decoding algorithm picks the most likely token v_n in h_τ. An example is shown in Figure 2.2. Formally, greedy decoding selects the most likely alignment $\mathbf{a}^*$, following Eq. (2.4):

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} P(\mathbf{a}|\mathbf{h}) \tag{2.4}$$

To generate the hypothetic text transcription $\tilde{\mathbf{y}}$, the repetitions and blank

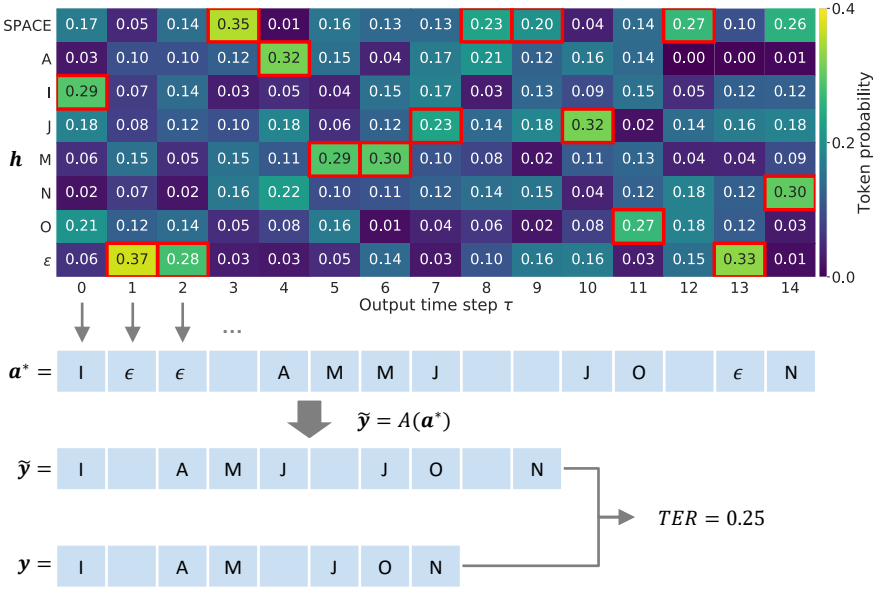


FIGURE 2.2: The CTC inference process with greedy decoding for a synthetic example with vocabulary $\mathbf{v} = \{SPACE, A, I, J, M, N, O, \epsilon\}$ and target text `I AM JON`. The network output $\mathbf{h}$ is decoded in a greedy fashion by picking the token with the highest probability at each time step τ, depicted by the red rectangle. The resulting alignment $\mathbf{a}^*$ is the alignment with maximum probability. The hypothetic text transcription $\tilde{\mathbf{y}}$ is obtained by removing repeated tokens and blank tokens ϵ. The hypothesis is compared to the target sequence $\mathbf{y}$ and yields a token error rate (TER) of 0.25 in this example.

labels are removed from $\mathbf{a}^*$ by the mapping function $\mathcal{A}$ as defined in Eq. (2.5)

$$\tilde{\mathbf{y}} = \mathcal{A}^{-1}(\mathbf{a}^*). \tag{2.5}$$

After decoding, the hypothesis $\tilde{\mathbf{y}}$ and target sequence $\mathbf{y}$ can be compared with the token error rate (TER) defined in Eq. (2.6):

$$TER = \frac{S + D + I}{N} \tag{2.6}$$

The numerator corresponds to the *Levenshtein distance* [70], also called *edit distance*. This distance measure is defined as the minimum number of substitutions S, deletions D or insertions I that are required to modify the hypothesis $\tilde{\mathbf{y}}$ to the target sequence $\mathbf{y}$. The denominator corresponds to the number of tokens N in the target sequence $\mathbf{y}$ and is used for normalization purposes. The TER may be interpreted as word error rate (WER) or character error rate (CER) when the tokens correspond to words or characters, respectively.

2.4 PROPERTIES

The CTC approach to ASR has three defining properties:

1. The alignment between input and output sequences is *monotonic*. When advancing to the next input time step, the output token is either kept the same or advanced to the next one. This is a reasonable assumption for the ASR task. Also, the monotonic alignment property is helpful for *online* (also referred to as *streaming*) speech recognition, because input time steps can be processed with low latency and immediately be discarded once the results have been sent to the decoder. Note that for non-monotonic sequence-to-sequence tasks, e.g. machine translation, CTC is not useful.

2. CTC sums over all possible, *valid* alignments. Note that the term valid only refers to the correct order of tokens, but not to the correct temporal position of these tokens in the speech signal. Some valid alignments may not be linguistically correct, e.g. when all tokens are assigned to the first few time steps of the speech signal instead of the time steps when they are actually spoken. On trained CTC models, these linguistically incorrect alignments usually have lower probability than alignments that seem linguistically more correct [28].

3. The CTC alogrithm makes a *conditional independence* assumption between output tokens as defined in Eq. (2.2). This is not a reasonable

assumption for ASR, because language modeling shows that previous tokens can be used to guess the next token. However, CTC models may be integrated with external language models during decoding, as shown in [18–20, 22]. Combining a CTC model with a separately trained language model would not comply with the strict end-to-end criteria of a single training stage and single neural network model. However, part of the literature sticks to the term end-to-end model even when an external language model is included (e.g. [20]).

Other end-to-end approaches include RNN-T and encoder-decoder models. Both variants remove the conditional independence assumption, and are therefore able to implicitly learn a language model. The RNN-T approach shares the monotonic alignment property with CTC, and also considers all valid alignments as CTC does. In contrast, the encoder-decoder model generates a single, non-monotonic alignment with an attention mechanism. Recent research has shown that RNN-T models outperform CTC models in offline and online ASR scenarios [21, 71]. On a final note, it is mentioned that the thesis contributions are compatible with both RNN-T and encoder-decoder approaches.

3

A CURRICULUM LEARNING METHOD FOR IMPROVED NOISE ROBUSTNESS IN AUTOMATIC SPEECH RECOGNITION

This thesis chapter adapts text from published work in [72] and presents general training methods for improving noise robustness in single-channel ASR. In particular, a novel training strategy presented that exploits the benefits of curriculum learning [48] for noise robust ASR. By first training the network on low SNR levels down to 0 dB and gradually increasing the SNR range to encompass higher SNR levels up to 50 dB SNR, the trained network shows improved noise robustness over conventional multi-condition training methods.

The remainder of this chapter is organized as follows: Section 3.1 presents several baseline methods and the curriculum learning method for improved noise robustness. The evaluation setup is detailed in Section 3.2, with results given in Section 3.3. The results are discussed in Section 3.4 and concluding remarks are presented in Section 3.5.

3.1 TRAINING METHODS FOR IMPROVED NOISE ROBUSTNESS

3.1.1 *Baseline*

The baseline method takes advantage of conventional multi-condition training [73] to increase the noise robustness of the network. Pink noise is added to a clean dataset to create samples with the desired SNR as depicted in Figure 3.1. Each training sample is randomly chosen to be of an SNR level in the range 0 to 50 dB with 5 dB steps. This wide range is larger than the SNR ranges used in previous work (e.g. 0 to 30 dB as in [40]). An extensive hyperparameter search showed that using such a large range resulted in the best performance on the test datasets. The noise mixing is done once at the waveform-level before filterbank audio features are computed. This one set of training data is presented to the network over all training epochs. The resulting network will be referred to as `NOISY` model. For completeness, a `CLEAN` model is included, i.e. a network that is only trained on clean speech.

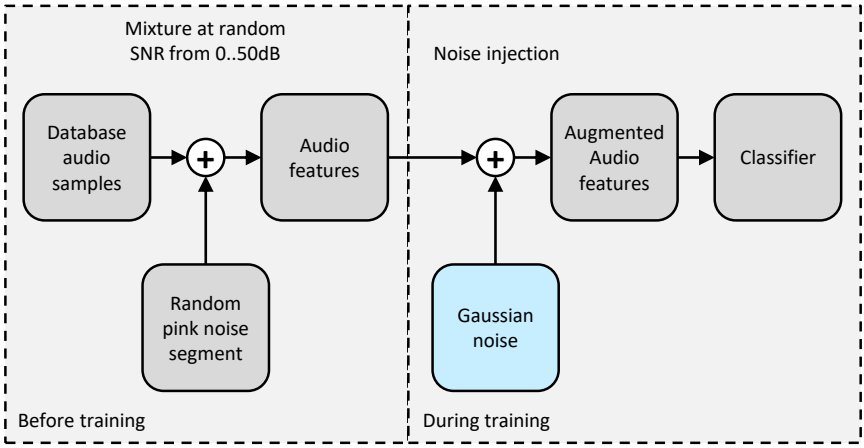


FIGURE 3.1: The baseline `NOISY` training method, depicted by grey boxes, first creates a noise corrupted version of the training set (before training), and then re-uses the same, noise-corrupted training set for every training epoch (during training). The `GAUSS` method adds Gaussian noise injection during training (blue box).

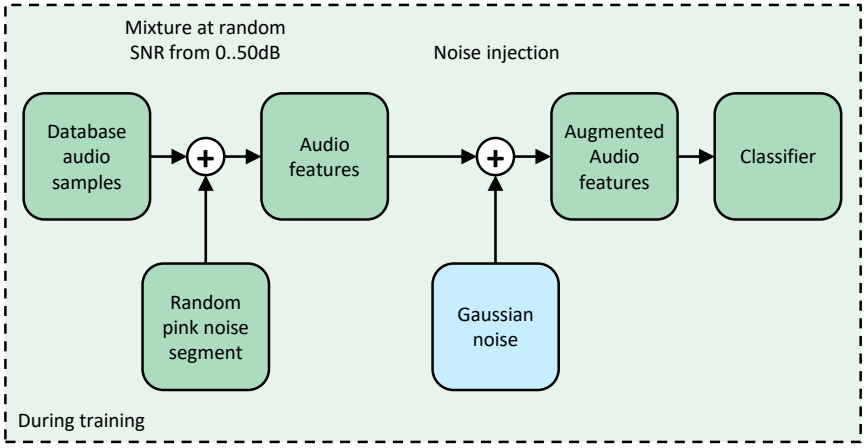


FIGURE 3.2: The `VANILLA-per-epoch noise mixing (PEM)` training method, depicted by green boxes, creates a newly mixed, noise corrupted training set for every training epoch. Therefore, each sample will be presented at different SNR levels and with different noise segments over the epochs. The `GAUSS-PEM` method adds Gaussian noise injection during training (blue box).

3.1.2 *Gaussian Noise Injection*

Gaussian noise injection is a well-known method for improving generalisation in neural networks [74]. It is used here to improve the noise robustness of the network.

During training, artificial Gaussian noise is added to the filterbank features created from the different SNR samples, as shown in Figure 3.1. The additive noise is drawn from a zero-centered Gaussian distribution $\mathcal{N}(\mu = 0, \sigma = 0.6)$. A hyperparameter search showed that using a Gaussian with a standard deviation of $\sigma = 0.6$ yielded the lowest error rates. The resulting network is referred to `GAUSS` model in the rest of the paper.

3.1.3 *Per-epoch Noise Mixing (PEM)*

The per-epoch noise mixing (PEM) is a method for adding noise to the waveform level during training. In every training epoch, each training sample is mixed with a randomly sampled noise segment at a randomly sampled SNR as depicted in Figure 3.2. The training procedure consists of the following steps:

1. Mix every training sample with a randomly selected noise segment from a large pink noise pool to create a resulting sample at a randomly chosen SNR level between 0 to 50 dB.

2. Extract audio features (e.g. filterbank features) for the noise-corrupted audio to obtain the training data for the current epoch.

3. *Optional*: add Gaussian noise drawn from the distribution $\mathcal{N}(\mu = 0, \sigma = 0.6)$ to the audio features.

4. Train on the newly generated training data from steps 1 to 3 for one epoch.

5. After the epoch is finished, discard this training data to free up storage.

6. Repeat from step 1 until training terminates.

The PEM method has several key advantages over conventional pre-training preprocessing methods. Firstly, it enables unlimited data augmentation on large speech datasets. With conventional methods, augmenting training data at the waveform level with real-world noise at various SNR values is prohibitively expensive in terms of processing time and training data size.

PEM allows use to overcome these restrictions by training on the GPU and pre-processing the next-epoch training data in parallel on the CPU. After an epoch was trained on, the training data gets discarded to free storage for the next epoch.

Secondly, PEM shows the network more unique training data: every training sample is presented at a selection of SNRs and with as many noise samples as can be extracted from the noise file and as needed by the number of epochs to reach a steady-state accuracy level. Thirdly, other noise types, different SNR training ranges, and even different audio features could be quickly tested as the training data can be easily augmented online. Finally, PEM allows to dynamically change the SNR level during training, which renders advanced training paradigms such as curriculum learning (Section 3.1.4) feasible.

In contrast to the plain Gaussian noise injection, the PEM method permits more control over the training data. Real-world noise is added to the acoustic waveform at controlled SNRs levels, ensuring that the training data corresponds to realistic noise corruption. The realism of these samples can be verified by listening tests. Of course, PEM can also be combined with Gaussian noise addition (optional step three). The networks trained with PEM only and PEM with Gaussian noise injection are referred to as `VANILLA-PEM` and `GAUSS-PEM` models, respectively.

3.1.4 *Curriculum Learning*

Neural networks have been shown to perform best when tested on the SNR levels they are trained on [73]. Under domain mismatch conditions, performance degrades: when tested on noisy conditions, a network trained on clean conditions fares worse than a network trained on noisy conditions. Also, networks trained on a vast SNR range generally do worse on a single SNR than networks optimized for this specific SNR. In order to achieve high accuracy under both high and low SNR with a single network, this work proposes a novel training paradigms based on curriculum learning. While curriculum learning has been used in image classification (scheduled denoising autoencoders, [75]) as well as speech recognition (SortaGrad [20], a method that sorts samples by sequence length for faster accuracy convergence), this is the first work targeted at improving the robustness of end-to-end ASR models under noisy conditions.

The novel accordion annealing (ACCAN) training method applies a multi-stage training schedule: in the first stage, the neural network is trained on the lowest SNR samples. In the following stages, the SNR training range

Model	Stage 1	Stage 2	Stage 3	...	Stage 10
`ACCAN`	0	0, 5	0, 5, 10	...	0, ..., 50
`ACCAN REV`	50	50, 45	50, 45, 40	...	50, ..., 0

TABLE 3.1: The SNR schedules for the `ACCAN` and `ACCAN REV` models: SNR ranges [dB] of the training stages © 2017 IEEE

is expanded in 5 dB steps towards higher SNR levels. A typical schedule is shown in Table 3.1. In every stage, training repeats until the WER on the development set no longer improves. At the end of each stage, the weights of the best network are stored and used as the starting point for the next stage. Both training and validation sets share the same range of SNR levels. The ACCAN approach seems counter-intuitive as noisy training data should be harder to train on than clean data. However, the noise allows the network to explore the parameter space more extensively at the beginning [48]. This study also includes a reversed ACCAN method which expands from high SNR to low SNR. The networks trained with ACCAN and reversed ACCAN will be referred to as the `ACCAN` and `ACCAN REV` models.

3.2 EXPERIMENTAL SETUP

AUDIO DATABASE All experiments were carried out on the Wall Street Journal (WSJ) corpus (LDC93S6B and LDC94S13B) [76] in the following configuration:

- training set: train-si84, 14 hours, 7138 samples,
- development set: test-dev93, 1 hour, 503 samples,
- test set: test-eval92, 0.7 hours, 333 samples.

For noise corruption, two different noise types were used: *pink noise* generated by the Audacity [77] software and *babble noise* from the NOISEX database [78].

DATA PREPARATION AND LANGUAGE MODEL The target text transcriptions were extracted with EESEN [22] routines. All experiments are character-based and use 58 labels (letters, digits, punctuation marks etc.) During test time, the network output was decoded with the Weighted finite

state transducer (WFST) approach from the EESEN framework, which leverages a tokenizer (map network output to character tokens), a lexicon (correct spelling mistakes) and a trigram language model (correct for most probable word sequences) to correct the network output text. The language model is trained on an expanded vocabulary in order to avoid out-of-vocabulary words occurring in the standard WSJ language model [22].

AUDIO FEATURES The audio waveforms were pre-processed to 123-dimensional filterbank features (25ms frames, 10ms frame shift, 40 Mel-spaced filterbanks, energy term, first and second order delta features). The features were generated by preprocessing routines from EESEN [22]. Each feature dimension is zero-mean and unit-variance normalized. Example filterbank features for clean and noisy conditions (pink noise and babble noise at 0dB SNR) are plotted in Figure 3.3.

NEURAL NETWORK CONFIGURATION The ASR pipeline is depicted in Figure 3.4 and uses a LSTM-based acoustic model combined with a WFST-based language model similar to the EESEN pipeline [22]. In order to automatically learn the alignments between speech frames and label sequences, the CTC [28] objective was adopted. The Lasagne library [79] was used to build and train a 5-layer neural network as the acoustic model. The first 4 layers consisted of bidirectional LSTM [80] units with 320 units in each direction. The fifth and final layer was a fully connected layer with 59 outputs, corresponding to the 58 character labels + one blank label required by CTC. The neural network contained 8.5M tunable parameters. All layers were initialized with the Glorot uniform strategy [81]. Every experiment started with the exact same weight initialization. During training, the Adam [82] stochastic optimization method was used with the standard learning rate $1e-3$. To prevent overfitting and to increase noise robustness, dropout [47] was used (dropout probability=0.3). Every epoch of training, the WER on the development set was monitored with a greedy decoding approach: at every frame, the most likely label was picked.

With all training strategies except ACCAN, the network was trained for a generous 150 epochs. The network weights from the epoch with the lowest WER were kept for evaluation. Generally, the improvements in WER saturated well before 150 epochs were reached. The ACCAN method used a patience of 5 as to switch between SNR stages, i.e. if the WER did not improve for 5 epochs on the current SNR stage, the training continued on the next SNR stage. By respecting the stage-switching policy, ACCAN reached

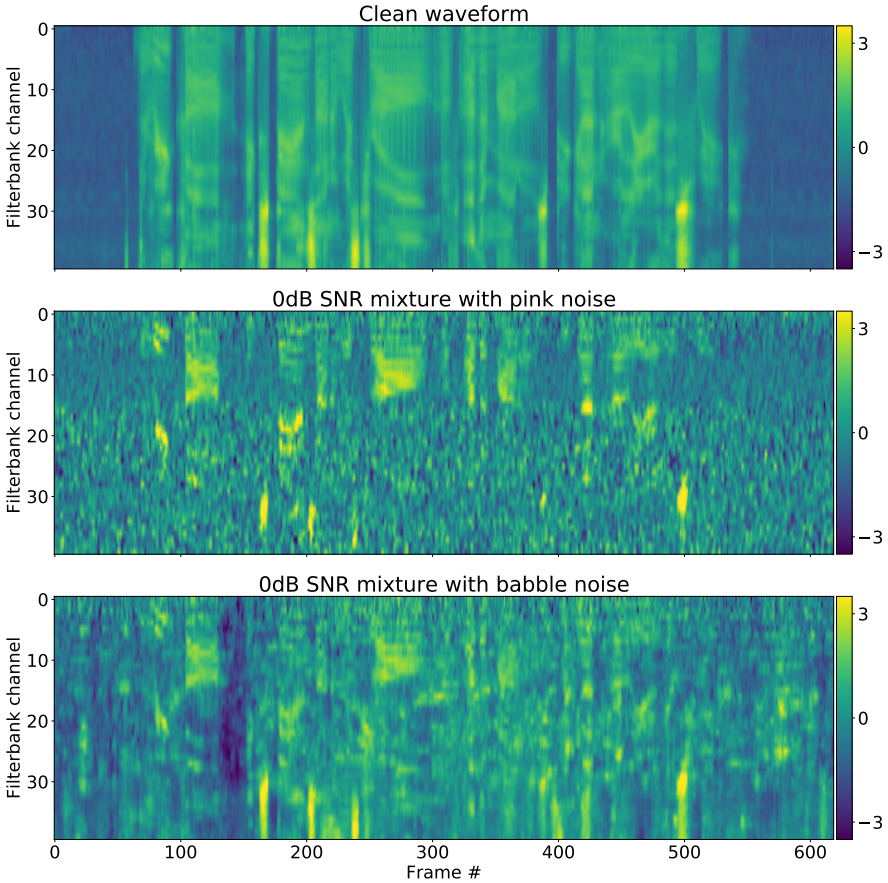


FIGURE 3.3: Filterbank features for a clean waveform from the WSJ test set (top) and when mixed at 0dB SNR with pink noise (middle) or babble noise (bottom). The features were normalized to zero mean and unit variance per each of the 40 filterbank channels. The 1st and 2nd order delta features are not plotted here.

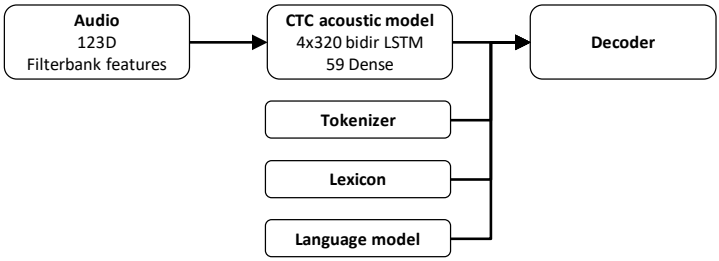


FIGURE 3.4: The ASR pipeline transcribes audio filterbank features to a text output. The CTC acoustic model emits sequences of characters that are decoded and corrected with a WFST-based tokenizer (remove CTC blank labels and repetitions), lexicon (correct for spelling errors) and trigram language model (select more likely word sequences).

the final SNR stage with the full SNR range at epoch 190. Saturation kicked in at epoch 240. While ACCAN trained for more epochs than the others, it only trained for 50 epochs on the full SNR range.

3.3 RESULTS

The results are reported for the test-eval92 subset of the WSJ dataset. The evaluation was carried out in 16 different conditions: clean condition and with added pink noise or babble noise at 15 SNR levels from 50dB to -20 dB in 5 dB steps. The results in Tables 3.2 and 3.3 report the average WER over the following SNR ranges:

- Full SNR range: [clean signal, 50dB to -10dB]
- High SNR range: [50dB to 0dB]
- Low SNR range: [0dB to -10dB]
- Range of interest (ROI): [20dB to -10dB]

The ROI is included, as subjective hearing tests showed that this range seems to well reflect common scenarios in public environments, where a clean speech signal is most often not available. Detailed results for each SNR individually are given in Table 3.4 for pink noise and Table 3.5 for babble noise. Results for -15dB and -20dB are reported too, but should be considered as extreme

cases. WER improvements are given as relative improvements in the text, and the relative WER compared to the baseline model CLEAN is reported in Figures 3.5 (a) and (b).

3.3.1 *Noise Addition Methods*

This section summarizes results from all models without curriculum learning: CLEAN, NOISY, GAUSS, VANILLA-PEM and GAUSS-PEM. The CLEAN model (trained on clean speech only) achieves 13.8% WER with a trigram language model and a 8.5M parameter network, while in literature [18], a 13.5% WER was reported using 3x larger, 26.5M parameter network and a similar decoding process. This confirms that the evaluated ASR pipeline is fully functional.

BASELINE CLEAN AND NOISY The model NOISY starts with a 25% higher WER on the clean test set than the model CLEAN. For SNRs lower than 25dB, the NOISY model is significantly more noise robust. The WER seems to drastically increase at 25dB for the CLEAN, while NOISY sees the increase onset at a lower 10dB SNR. However, all other methods outperform the models NOISY and CLEAN by a significant margin at high and low SNRs.

VANILLA-PEM VS. GAUSS: Compared to the baseline NOISY model, VANILLA-PEM achieves a 23% decrease in WER on high SNR, while GAUSS only reduces WER by 15% (both pink noise and babble noise). This results in VANILLA-PEM being able to outperform the baseline CLEAN model on clean speech, while GAUSS is not able to do the same. On low SNR, both models reduce WER by around 20% on the pink noise test set. On babble noise, PEM achieves a higher 22.5% WER decrease compared to the 15.5% decrease provided by GAUSS.

GAUSS-PEM: The GAUSS-PEM model achieves the overall lowest WER on the high and low SNR range. It beats the baseline NOISY model by between 26.5% and 28.7% on high SNR, on low SNR and on the ROI for both pink noise and babble noise. The results on the high SNR range are notable: GAUSS-PEM is able to outperform the baseline CLEAN network at every single SNR step in the high SNR range, even on clean speech. The GAUSS-PEM model is therefore much more noise robust while at the same time it even improves clean speech scores. Around 35dB to 25dB, GAUSS-PEM (other models, too) reaches its minimum WER. This is expected, as the mean SNR of the training

TABLE 3.2: Average absolute WER [%] for given SNR ranges after decoding, pink noise conditions. Printed bold: lowest WER. © 2017 IEEE

	SNR range			
Model	Full	High	Low	ROI
CLEAN	54.7	29.0	109.6	67.9
NOISY	46.0	23.3	88.6	51.7
GAUSS	37.4	19.8	71.1	42.1
VANILLA-PEM	35.6	17.8	70.6	40.8
GAUSS-PEM	**34.1**	**16.6**	64.7	37.2
ACCAN	34.4	18.1	**59.5**	**36.0**
ACCAN REV	35.2	17.8	66.3	38.8

TABLE 3.3: Average absolute WER [%] for given SNR ranges after decoding, babble noise conditions. Printed bold: lowest WER. © 2017 IEEE

	SNR range			
Model	Full	High	Low	ROI
CLEAN	53.0	32.0	113.7	72.1
NOISY	53.3	29.9	114.0	68.4
GAUSS	45.4	25.4	96.3	56.9
VANILLA-PEM	41.0	22.8	88.3	52.3
GAUSS-PEM	**39.5**	21.6	83.7	49.0
ACCAN	39.6	**21.5**	**80.2**	**47.0**
ACCAN REV	**39.5**	**21.5**	82.9	48.2

SNR [dB]	Model						
	CLEAN	NOISY	GAUSS	VANILLA PEM	GAUSS PEM	ACCAN	ACCAN REV
Clean	13.8	17.3	15.7	**13.3**	13.6	15.9	14.6
50	14.4	17.4	15.8	**13.2**	13.5	15.8	14.4
45	14.0	17.3	15.7	**13.2**	13.6	15.4	14.3
40	13.8	16.9	15.6	**12.7**	13.4	15.3	14.1
35	13.7	16.5	14.8	**12.6**	13.2	15.0	13.3
30	13.7	16.4	14.4	**12.6**	**12.6**	15.0	13.4
25	16.1	16.2	14.5	12.9	**12.4**	15.2	13.9
20	18.9	16.8	15.3	13.6	**12.8**	15.9	14.4
15	25.9	19.0	16.9	15.1	**14.2**	16.1	15.4
10	40.1	23.4	20.2	18.9	**17.0**	18.5	18.5
5	61.8	36.5	28.9	26.2	**22.3**	22.9	24.4
0	86.4	59.8	45.5	45.0	37.6	**33.7**	40.2
-5	109.0	90.0	72.8	73.5	66.3	**58.8**	67.8
-10	133.4	116.2	94.9	93.4	90.2	**85.9**	90.9
-15	147.2	126.7	99.0	96.9	95.9	**95.6**	97.0
-20	152.8	129.5	98.9	97.1	96.8	**96.2**	97.2

TABLE 3.4: Testing against *pink noise*: absolute WER [%] on single SNRs after decoding. Printed bold: lowest WER.

	Model						
SNR [dB]	CLEAN	NOISY	GAUSS	VANILLA PEM	GAUSS PEM	ACCAN	ACCAN REV
Clean	13.8	17.3	15.7	**13.3**	13.6	15.9	14.6
50	14.2	17.1	15.6	**13.2**	13.8	15.7	14.4
45	14.2	16.9	15.7	**12.9**	13.7	15.3	14.2
40	13.9	16.7	15.7	**12.7**	13.4	14.9	14.0
35	14.2	16.1	15.3	**12.3**	13.4	15.1	14.1
30	14.5	15.7	15.0	**12.7**	13.3	15.1	14.0
25	15.7	15.8	15.4	**12.8**	13.7	15.0	14.0
20	18.8	17.8	16.5	**14.0**	14.6	15.5	14.6
15	26.6	23.1	19.5	17.4	16.9	17.5	**16.5**
10	43.9	35.5	27.5	25.6	22.9	**21.8**	21.9
5	74.2	60.6	45.9	44.2	37.4	**33.4**	35.5
0	102.2	94.1	77.4	72.7	64.1	**57.2**	63.5
-5	116.6	119.4	102.6	93.2	89.8	**86.1**	88.7
-10	122.4	128.4	109.0	99.1	97.3	97.2	**96.6**
-15	122.3	129.3	109.8	99.5	**97.3**	98.8	97.6
-20	121.4	129.2	110.6	99.8	**97.4**	99.1	97.6

TABLE 3.5: Testing against *babble noise*: absolute WER [%] on single SNRs after decoding. Printed bold: lowest WER.

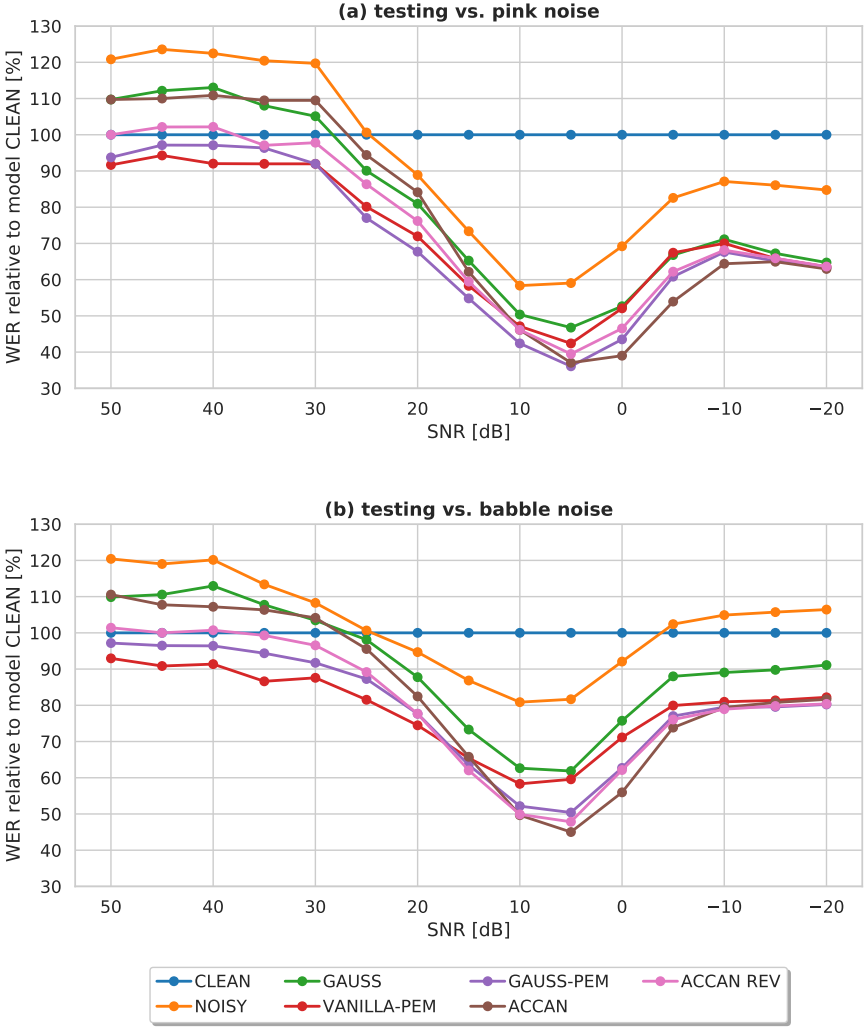


FIGURE 3.5: The WER [%] relative to the `CLEAN` model when testing vs. (a) pink noise and (b) babble noise. The `ACCAN` model achieves the lowest error rate of all models from 0 to -10 dB SNR for pink noise, and for 5 to -5 dB for babble noise.

SNR range is 25dB and the network seems to optimize for SNR levels close to this value [73].

3.3.2 *Curriculum Learning*

To further increase the noise robustness, the novel curriculum learning strategy ACCAN was proposed. The results for the `ACCAN` model are compared to the `GAUSS-PEM` model, as this was the most noise robust non-curriculum learning model. The evaluation shows increased noise robustness for `ACCAN` on pink noise and babble noise: the WER decreases between 3.3% (ROI, pink noise) and 4.1% (ROI babble noise). For pink noise, the biggest decrease is seen at 0dB (10.5% WER decrease) and -5dB (11.3% decrease). For babble noise, the biggest WER decreases are found at 10dB (4.9%), 5dB (10.9%) and 0dB (10.7%).

The average WER of the `ACCAN` model on the high SNR range is worse on pink noise (relative 8.8% increase in WER), but better on babble noise (relative 0.4% decrease in WER) when compared to `GAUSS-PEM`. Ultimately, the absolute WER in clean speech of `ACCAN` (15.9%) is better than the baseline `NOISY` model (17.3%) but worse than the `GAUSS-PEM` model (13.6%).

3.4 DISCUSSION

All proposed training methods lead to networks with increased noise robustness in the low SNR range in comparison with the standard `NOISY` baseline model. The noise robustness is increased on a network level and it does not rely on complex preprocessing frameworks.

The increased noise robustness is seen against both pink and babble noise types. This is remarkable, as the networks only saw the pink noise type during training. The results show that waveform-level noise mixing as used in PEM is especially strong in transferring noise robustness to noise types not seen in training. The feature level Gaussian noise injection is less effective on unseen noise types. Also, PEM enabled us to train noise robust networks that - at the same time - achieve lower WER on clean speech than a network trained only on clean speech. The uncompromising data augmentation by PEM should be a decisive factor to achieve these results. For example, while the baseline `NOISY` model was trained on 1.7GB of unique data (waveform level), the PEM-enabled models trained on up to 408GB (240 epochs for `ACCAN` * 1.7GB) of unique training data (waveform level). By permanently sampling different noise segments, the network is forced to not rely only on

constant noise features for classification, but to develop a better internal representation of the speech data. This representation could be refined further by using other noise types besides pink noise for training, such as babble noise, street noise, restaurant noise. Also, SNR steps smaller than 5dB could be used to allow more than 11 different SNR levels during training.

The PEM methods allows to dynamically change the SNR level during training. This facilitated the implementation of the novel ACCAN training strategy, that achieved the best noise robustness performance. The multi-stage training starts at low SNRs levels, where annealed networks are able to explore the parameters space with moderate influence of the speech signal. During gradual exposure to higher SNRs in the training process, accordion annealed networks refine their internal model of speech step by step, while they seemingly acquire higher noise robustness at the lower SNR levels. The inverse way of going through the SNR range, i.e. high to low SNR, did not yield increased noise robustness. The immediate presence of clean speech signals may have forced the network to converge faster to a more complex acoustic model instead of exploring the parameter space.

3.5 CONCLUSION

This thesis chapter proposed new training methods for improving the noise robustness of end-to-end ASR models for a large-vocabulary continuous speech recognition (LVCSR) task. The networks were trained for a wide SNR range with the use of the PEM training method which adds noise at the waveform level and the Gaussian noise injection method which adds Gaussian noise at the feature level. By combining the Gaussian noise injection and PEM methods into the Gauss-PEM method, an average 28% WER reduction was achieved on the 20dB to -10 dB SNR range when compared to a conventional multi-condition training method. At the same time, the WER was lower on clean speech than for a network that was trained solely on clean speech. The ACCAN training strategy enhanced the Gauss-PEM method with a curriculum learning strategy and resulted in performance up to 11.3% lower WER at low SNRs compared to Gauss-PEM method.

In the larger context of this thesis, this initial study showed that there is tremendous potential to increase the noise robustness of end-to-end models. However, the restrictions to single-channel input and no architectural changes were rather arbitrary, and present a potential limitation towards further reducing error rates. Also, the overall goals of this thesis are not only reducing error rates, but also improving the interpretability of end-to-end models. From

a critical point of view, the proposed training methods mainly increased the amount of training data, without making the model more interpretable. The next thesis chapter will remove the mentioned restrictions by considering multi-channel input and architectural changes, and also attempt to improve the model interpretability.

4

ATTENTION-DRIVEN MULTI-SENSOR SELECTION

This thesis chapter explores end-to-end models in the context of multi-channel ASR and multi-modal AVSR. To deal with multiple input sensors, a sensory attention mechanism is proposed that is embedded in a sensor transformation attention network (STAN). STANs support multi-sensor inputs of the same or different modalities in a single end-to-end framework. The sensory attention mechanism enables STANs to tune their attention towards less noisy sensors for improved accuracy in noisy conditions. The attentional signal is highly interpretable and correlates with the sensor noise level.

The remainder of this chapter is organized as follows. First, the STAN framework and the sensory attention mechanism are described (Section 4.1). The evaluation covers multi-modal AVSR with synthetic noise (Section 4.2), and multi-channel ASR with real world noise, using either filterbank features (Section 4.3) or spectrogram features (Section 4.4) as input. A final conclusion closes this chapter (Section 4.5). The text in this chapter has been adapted from published work in [83] (Sections 4.1, 4.2, 4.3) and in [84] (Section 4.4).

4.1 MULTI-SENSOR ATTENTION MODEL

The STAN architecture depicted in Figure 4.1 includes the following building blocks: (1) input sensors, (2) sensor transformation functions and (3) a sensory attention mechanism. The attention mechanism combines multiple input sensors into a single, merged representation by first weighting and then summing transformed feature frames from individual sensors.

We assume a multi-sensor setup with $i = 1, ..., N$ sensors. All sensors record time series that are binned into $k = 1, ..., K$ frames, such that every sensor s^i provides a D_f-dimensional feature vector $f_k^i \in \mathbb{R}^{D_f}$ for each frame k. The merged representation $m \in \mathbb{R}^{D_t}$ is generated by the steps described in Eq. (4.1) to (4.4):

$$t_k^i = T^i(f_{1..k}^i) \tag{4.1}$$

$$z_k^i = Z^i(t_{1..k}^i) \tag{4.2}$$

$$\alpha_k^i = \frac{\exp(z_k^i)}{\sum_{j=1}^{N} \exp(z_k^j)} \tag{4.3}$$

$$m_k = \sum_{i=1}^{N} \alpha_k^i \cdot t_k^i \tag{4.4}$$

The transformation function T^i converts the feature vectors f_k^i to transformed feature vectors $t_k^i \in \mathbb{R}^{D_t}$ (Eq. 4.1). If no transformation is desired, then T^i is the identity function $f_k^i = t_k^i$. The attention scoring function Z^i produces scalar attention scores $z_k^i \in \mathbb{R}^1$ based on the transformed features of sensor i (Eq. 4.2). The attention weights $\alpha_k^i \in \mathbb{R}^1$ are computed by performing a softmax operation over all attention scores $z_k^i \in \{z_k^1, ..., z_k^N\}$ (Eq. 4.3), and therefore $\sum_{i=1}^{N} \alpha_k^i = 1$. Each transformed feature vector t_k^i is then scaled by the corresponding attention weight α_k^i and merged through a summation operation (Eq. 4.4). The resulting - transformed, scaled and merged - feature vectors m_k are then presented to the classifier.

The sensory attention model implements the scoring functions Z^i which can be modelled using neural networks. In this study, the scoring function Z^i is implemented with LSTM units [1] [80] followed by one dense unit (weight

1 Any number from 10 to 200 LSTM units perform equally well in preliminary experiments.

W, bias b) with a SELU non-linearity [2] [85] (Eq. 4.5). LSTM units are a convenient choice because past history is automatically considered.

$$Z^i(t^i_{1..k}) = \text{SELU}(W \cdot \text{LSTM}(t^i_{1..k}) + b) \tag{4.5}$$

The proposed sensory attention mechanism has the following useful properties:

1. As a *soft* attention mechanism, it is differentiable and trainable with back-propagation, and therefore compatible with end-to-end models.

2. At each frame k, the attention weights α^i_k sum up to 1 across all sensors. Therefore, the attention weight indicates the contribution of single sensors to the combined representation of a frame.

3. The attention scores (z^i_k) and weights (α^i_k) are computed on every frame. Their values reflect the dynamic per frame adjustment for temporal changes in signal quality due to noise, sensor failure or other sensor corruptions.

4. The attention scoring function of each sensor is independently evaluated, and existing sensors may be removed or new sensors may be added after training.

5. The scoring function Z^i of each sensor may be identical when their parameters are shared ($\theta_{Z^1} = \theta_{Z^2} = ... = \theta_{Z^N}$). In the shared case, the attention mechanism would then be invariant to sensor re-ordering because the same scoring function is used for all sensors.

The same arbitrary choice of functions can be made for the transformation functions T^i as in the scoring functions Z^i. This study proposes dense units or identity functions. However, other network types such as CNNs [86] might work just as well or even better.

2 The choice of the SELU activation function is arbitrary. In preliminary studies, different variants (ReLU, LeakyReLU, SELU) were found to work equally well. The normalization effect from SELU is not required.

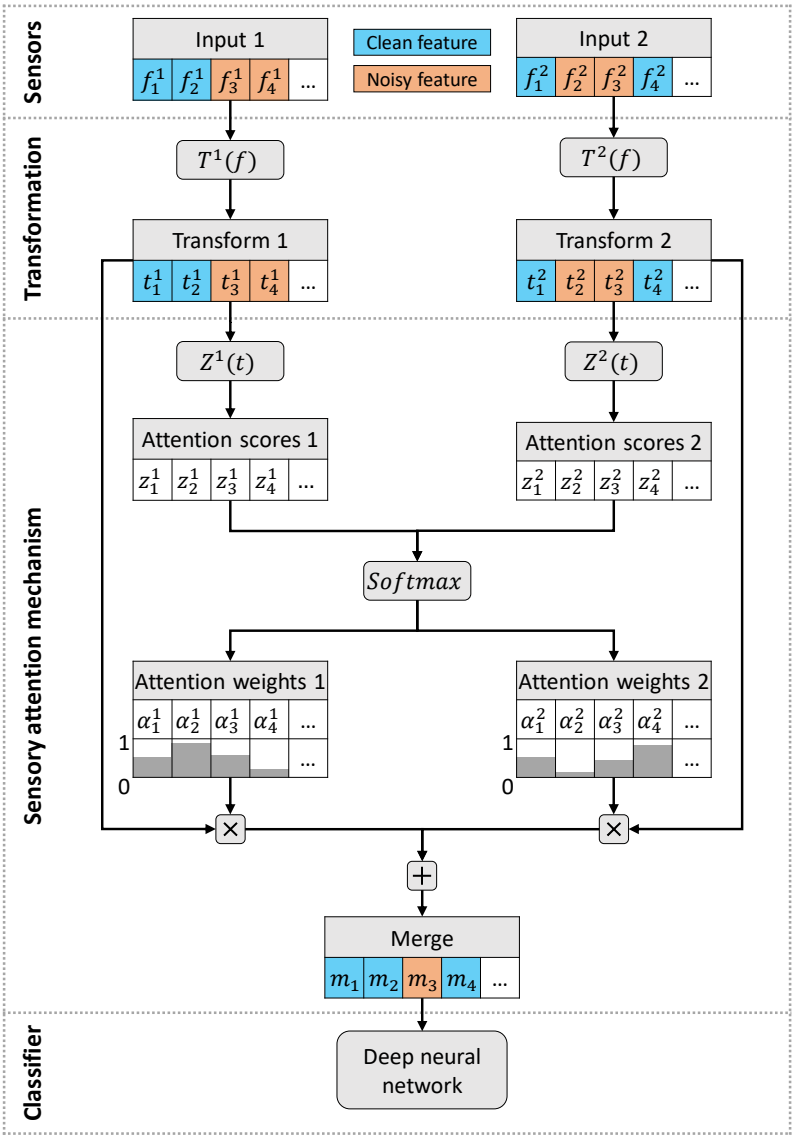


FIGURE 4.1: STAN architecture for a setup with two input sensors. The input feature vectors f_k^i are transformed and then weighted and summed to generate the merged representation m_k that is used for classification. The sensory attention mechanism dynamically adapts its attention weights to create a cleaner merged representation. © 2019 IEEE

4.2 EXPERIMENTS WITH MULTI-MODAL INPUT

The impact of non-stationary noise on the attention mechanism in a multi-modal setting is evaluated on the audio-visual GRID dataset. These experiments use controlled noise levels to establish the ground truth in measuring the correlation between the attention weights and the SNR of the input sensors.

4.2.1 *Dataset*

The GRID dataset [87] provides audio and video (facial) recordings of 1000 sentences each spoken by 34 speakers. The recording setup consisted of a single camera and microphone. Each sentence contains 6 vocabulary units out of a word vocabulary of 51 classes (commands, colors, prepositions, adverbs, letters and digits). The samples were shuffled and split by 80/10/10% into training, validation and test sets. The raw audio was converted to 123 dimensional filterbank features (40 Mel-spaced filterbanks, energy coefficient, 1st and 2nd order delta features, 50ms frames, 25ms frame shift). The video recordings were processed to extract 17x8 pixel-sized mouth crops by using the Dlib face detector and a pre-trained model of the 68 facial landmark annotator [88, 89]. An example of the audio and video input features is presented in Figure 4.2. The pre-processed features of both modalities were presented to the networks with the same frame rate and no further temporal alignment efforts were applied. Both feature types were zero-mean and unit-variance normalized on a per-sample basis. The task in the GRID experiments is AVSR: sequences of audio and video input features are transcribed to sequences of words. The WER was used as the performance metric.

4.2.2 *End-to-end Models*

Five models are evaluated: the uni-modal, single-sensor (1) `AUDIO` and (2) `VIDEO` models and the multi-modal, two-sensor (3) `CONCAT`, (4) `AVG` and (5) `STAN` models. All models convert each of their input sensors with individual transformations T^i (Eq. 4.1) that are implemented with 50 dense units followed by a SELU non-linearity [85], therefore $t_k^i \in \mathbb{R}^{50}$. The multi-modal networks combine the transformed features t_k^i from each sensor by concatenation (`CONCAT`), averaging (`AVG`) or with the sensory attention mechanism (`STAN`). The `STAN` attention scoring functions $\{Z^1, Z^2\}$ are both implemented with 20 LSTM units followed by a single dense unit with a SELU non-linearity

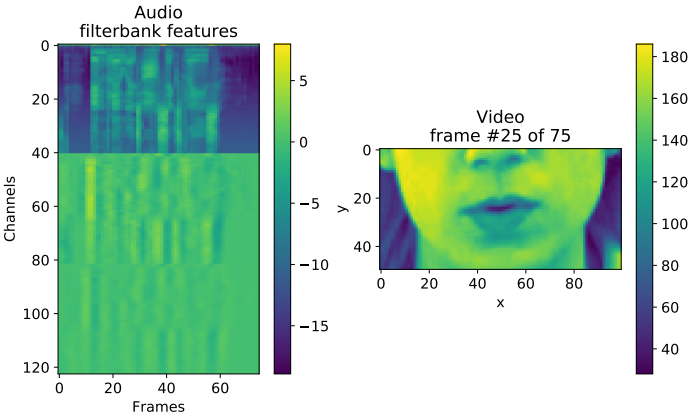


FIGURE 4.2: The input features for a randomly selected sample from the GRID multi-modal experiments. The left plot shows the 123D filterbank features for the audio modality before normalization. The right plot shows the mouth-cropped video frame before normalization and downscaling. All 75 frames of the sequence are plotted for audio, and only the 25th frame of the sequence is plotted for video.

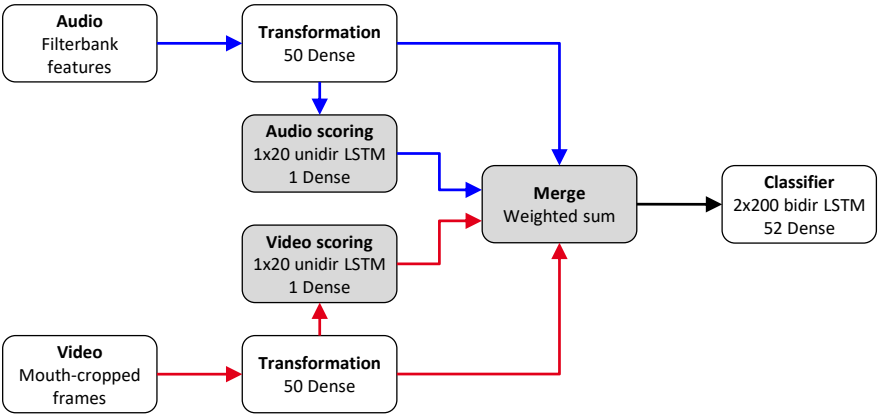


FIGURE 4.3: The `STAN` model architecture for multi-modal input on the GRID experiments. The grey boxes correspond to elements that are part of the sensory attention mechanism. The blue and red arrows show the audio and video pathways, respectively. The scoring functions use relatively fewer units and parameters than the classifier. The `CONCAT` and `AVG` models would replace the sensory attention mechanism by concatenation and averaging operations respectively.

(Eq. 4.5), see Figure 4.3. Because the sensors provide distinct feature modalities, the parameters of the scoring functions are not shared, i.e. $\theta_{Z^1} \neq \theta_{Z^2}$. All models use the same classifier architecture consisting of 2 layers of bidirectional LSTM cells with 200 units per direction followed by a 52-dimensional output projection, but are individually optimized. The models are trained in an end-to-end fashion with the CTC objective [28] and the ADAM optimizer [82] for 150 epochs. The model achieving the lowest WER on the validation set is used for evaluation. Note that there is no modality-specific pre-training, and instead all models are trained from scratch with randomly initialized weights.

4.2.3 *Noise Models*

The clean data provided by either modality provides reasonable information to solve the ASR task on GRID, and a `STAN` model with clean multi-modal input did not show much attentional switching in consequence. To encourage attention switching, synthetic noise is added to each feature frame f_k^i. The noise is sampled from a zero-centered uniform distribution with standard deviation $\sigma(k)$, that is referred to as the noise level. Three different noise models are used: (1) random walk noise, (2) cross noise and (3) hi-lo noise, with examples shown in Figure 4.4.

The *random walk* noise model adds noise with a time-varying noise level, $\sigma(k)$, to each sensor and is used for both training and testing. The random walk process $q(k)$ is drawn separately for each sensor, and $q(k)$ is then normalized to the range $[0, 1]$ and scaled by the maximum noise level σ_{max} as shown in Eqs. (4.6) and (4.7). As a result, each sensor has its own distinct noise level at each time step.

$$q(k) = \sum_{i=1}^{k} n_i, \text{ with } n_i \sim \mathcal{N}(0, 1) \tag{4.6}$$

$$\sigma(k) = \sigma_{max} \cdot \frac{q(k) - \min\{q_1, ..., q_K\}}{\max\{q_1, ..., q_K\} - \min\{q_1, ..., q_K\}} \tag{4.7}$$

The resulting random walk process yields an average noise level of $E(\sigma(k)) = \sigma_{max}/2$. The training noise level $\sigma_{max} = 8$ is chosen such that the average WER is close to 16% when using a single modality. This allowed the multi-sensor models to have a good chance of improvement by considering the other modality, while at the same time a single modality still provided reasonable WER with 5 correct words out of 6. This noise level was necessary to encourage attentional switching for STANs.

The *cross* and *hi-lo* noise variants are only used during testing to evaluate the generalization of the attention mechanism. The cross noise applies a linearly increasing noise level $\sigma(k) = \sigma_{max} \cdot k/K$ to one sensor, and a linearly decreasing noise level $\sigma(k) = \sigma_{max} \cdot (1 - k/K)$ to the other sensor. The hi-lo noise applies a constant noise level $\sigma(k) = \sigma_{max}$ to one sensor, and the noise level $\sigma(k) = 0$ to the other sensor. The sensor that sees the increasing (cross noise) or maximum (hi-lo) noise level is alternated for every test sample. In consequence, either sensor sees the increasing or maximum noise level for 50% of all test samples. All models are tested with the same alternation pattern.

4.2.4 *Attention Metrics*

Because the noise level of both sensors is known at any time, the interpretability of the attention weights can be quantified. One proposed metric, attention correlation (ATTCORR), measures the correlation between the noise level and the attention weights for a specific sensor. The sensor index i (audio=1, video=2), noise level σ^i and attention weights α^i are used to define the ATTCORR as a correlation coefficient with values between -1 and 1:

$$\text{ATTCORR}^i = corr\Big(\big(1 - 2 \cdot \frac{\sigma^i}{\sum_{j=1}^{2} \sigma^j}\big), \big(2 \cdot \alpha^i - 1\big)\Big) \tag{4.8}$$

A value of 1 corresponds to perfect correlation, and a value of 0 corresponds to chance level. Another quantitative metric is the attention accuracy (ATTACC) which measures the accuracy of identifying the higher (or lower) SNR sensor by their attention weights:

$$\text{ATTACC}\,[\%] = 100 \cdot \frac{f_{correct}}{f_{total}} \tag{4.9}$$

where $f_{correct}$ is the number of frames with correct SNR sensor identification and f_{total} is the total number of frames of the evaluation set. Sensors are considered as correctly identified on a frame when the lower SNR sensor is attributed a lower attention weight. An ATTACC value of 100% corresponds to perfect identification, and a value of 50% corresponds to chance level.

4.2.5 *Results*

The models are trained in noisy conditions with added random walk noise ($\sigma_{max} = 8$), and evaluated in clean and noisy conditions with random walk,

cross or hi-lo noise added. Table 4.1 reports the WER for all five models; and the ATTACC and ATTCORR scores for the `STAN` model.

The multi-modal networks perform significantly better than the uni-modal networks in both clean and noisy conditions. The `STAN` and `AVG` models are mostly on par and achieve the lowest WER, except for the hi-lo noise where `STAN` shows a relative WER improvement of up to 36.1% ($\sigma_{max} = 8$) over `AVG`. Across all noise types and all noise levels, `STAN` shows a relative WER improvement of 9.1% to 36.9% over `CONCAT`, 56.8% to 77.7% over `AUDIO` and 64.1% to 82.8% over `VIDEO`.

The `STAN` model computes highly interpretable attention weights as shown in Figure 4.4. Both ATTACC and ATTCORR metrics start at chance level for $\sigma_{max} = 1e-5$ and increase to scores between 75.3% to 99.8% (ATTACC) and 0.68 to 0.83 (ATTCORR) for $\sigma_{max} = 8$. Failures in correct prediction of the lower SNR sensors mainly arise when both sensors have similar noise levels, or noise levels change too rapidly. These cases are mostly seen for the random walk noise. Even though the model is trained only on random walk noise, the attention mechanism generalizes well to the cross and hi-lo noise types.

4.2.6 *Discussion*

This initial study evaluated STANs in a noisy AVSR task. The attention mechanism increased the total number of network parameters by only a small amount, and enabled the STAN model to robustly process multi-modal input from audio and video sensors. Across all experiments, STANs achieved error rates on par or better than single-sensor models or multi-sensor models with concatenation and averaging strategies. The attention mechanism improved the noise robustness by dynamically tuning towards the higher SNR sensor, and the tuning process yielded highly interpretable attention weights that correlate with the sensor SNR levels. These results align well with the overall thesis goals of increasing noise robustness and model interpretability.

One particular design decision in this study was the use of synthetic noise. While synthetic noise does not reflect real-world scenarios, it allowed to establish the ground truth of sensor noise levels, and therefore to quantify the correlation between noise level and attention weights. Given the promising results on synthetic noise, the next sections will also evaluate STANs on real-world noise.

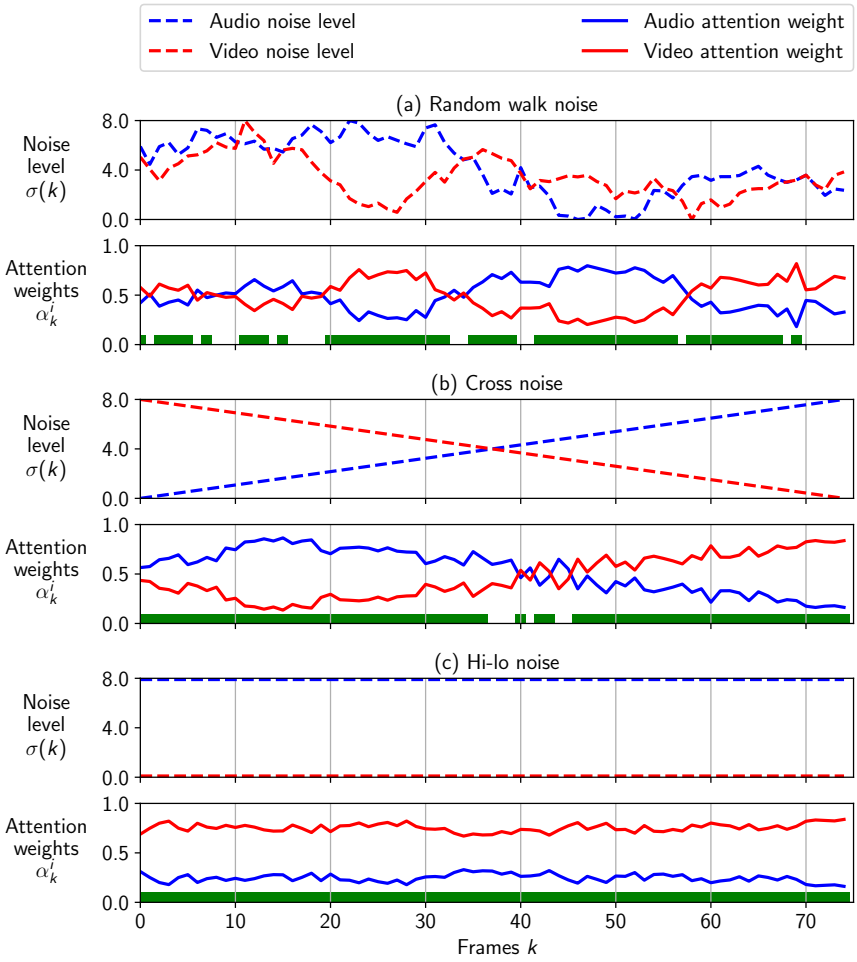


FIGURE 4.4: Attention response on a randomly selected sample from the multi-modal GRID dataset. The top row depicts the noise levels applied to each input, and the bottom row depicts the attention weights computed by **STAN**. The green bars indicate frames where the relative SNR value of the correct sensor is identified. (a) shows the response to random walk noise resulting in ATTACC of 72%. Note how the attention weights dynamically change, mostly in correlation with the noise level. (b) and (c) show responses to cross and hi-lo noise, with ATTACCs of 92% and 100% respectively.

Metric	min, max, chance level	Model	Random walk noise, σ_{max} 1e-5	4	8
WER [%]	0, 100, 98	AUDIO	2.8 ± 0.1	7.9 ± 0.3	18.5 ± 0.5
		VIDEO	5.2 ± 0.5	9.4 ±0.7	22.3 ± 0.9
		CONCAT	1.1 ± 0.2	2.9 ± 0.6	9.4 ± 1.2
		AVG	**1.0 ± 0.1**	**2.3 ± 0.2**	8.2 ± 0.4
		STAN	**1.0 ± 0.1**	2.4 ± 0.1	**8.0 ± 0.2**
ATTACC [%]	0, 100, 50	STAN	50.6 ± 0.1	59.7 ± 1.4	75.3 ± 1.1
ATTACC/scale	-100, +100, 0	STAN	1.2 ± 0.1	19.4 ± 2.8	50.6 ± 2.2
ATTCORR[1]	-100, +100, 0	STAN	1.2 ± 0.1	33.7 ± 2.8	68.0 ± 1.2
ATTCORR[2]	-100, +100, 0	STAN	1.3 ± 0.1	33.8 ± 2.7	68.2 ± 1.2

Metric	min, max, chance level	Model	Cross noise, σ_{max} 1e-5	4	8
WER [%]	0, 100, 98	AUDIO	2.8 ± 0.1	7.5 ± 0.3	19.1 ± 0.6
		VIDEO	5.2 ± 0.5	9.2 ± 0.7	22.0 ± 0.8
		CONCAT	1.1 ± 0.2	2.6 ± 0.5	8.8 ± 1.2
		AVG	**1.0 ± 0.1**	**2.1 ± 0.2**	7.4 ± 0.4
		STAN	**1.0 ± 0.1**	2.2 ± 0.1	**7.1 ± 0.3**
ATTACC [%]	0, 100, 50	STAN	49.2 ± 0.1	65.1 ± 2.1	87.6 ± 1.5
ATTACC/scale	-100, +100, 0	STAN	-1.6 ± 0.1	30.2 ± 4.2	75.1 ± 3.0
ATTCORR[1]	-100, +100, 0	STAN	-0.5 ± 0.1	53.6 ± 3.2	83.4 ± 1.8
ATTCORR[2]	-100, +100, 0	STAN	-0.5 ± 0.1	53.6 ± 3.2	83.4 ± 1.8

Metric	min, max, chance level	Model	Hi-lo noise, σ_{max} 1e-5	4	8
WER [%]	0, 100, 98	AUDIO	2.8 ± 0.1	10.7 ± 0.3	23.8 ± 0.5
		VIDEO	5.2 ± 0.5	13.3 ± 0.6	30.8 ± 0.4
		CONCAT	1.1 ± 0.2	3.2 ± 0.6	8.4 ± 1.2
		AVG	**1.0 ± 0.1**	2.8 ± 0.2	8.3 ± 0.7
		STAN	**1.0 ± 0.1**	**2.6 ± 0.1**	**5.3 ± 0.2**
ATTACC [%]	0, 100, 50	STAN	50.0 ± 0.1	87.3 ± 2.4	99.8 ± 0.1
ATTACC/scale	-100, +100, 0	STAN	0.0 ± 0.1	74.6 ± 4.8	99.7 ± 0.1
ATTCORR[1]	-100, +100, 0	STAN	-	-	-
ATTCORR[2]	-100, +100, 0	STAN	-	-	-

TABLE 4.1: Results of the GRID experiments, averaged over 10 runs. All values are reported in the format *mean* ± *standard deviation*. The ATTCORR values are not computed for the hi-lo noise because the correlation function is not defined for constant functions. The lowest WER is printed bold. The ATTCORR and ATTACC values are rescaled to the range $[-100, 100]$ in the interest of readability.

4.3 MULTI-CHANNEL SPEECH RECOGNITION WITH NATURAL NOISE

The STAN architecture is further evaluated on the multi-channel CHiME-4 dataset which includes *real-world* noise. The reported experiments include a comparison of STANs against concatenation, averaging and beamforming models. Without any re-training, the experiments also evaluate the robustness of the STAN models with respect to reversed channel orders, channel addition and channel removal.

4.3.1 *Dataset*

The CHiME-4 dataset [31] for ASR provides real and simulated noisy speech data from a tablet device with 6 microphones. The data was recorded in four noisy environments: public transport, a cafe, a street junction and pedestrian area. The real data was recorded with the tablet device, and the simulated data was generated by mixing clean speech utterances from the WSJ0 [76] dataset with environment background recordings. The tablet device is equipped with 5 microphones facing the speaker and 1 microphone facing away from the speaker (backward channel #2, the noisiest of all). Bothreal data (tr05_real, 1600 samples) and simulated data (tr05_simu, 7138 samples) are used for training, and the evaluation is carried out on the real noisy data subsets (et05_real, 1320 samples and dt05_real, 1640 samples).

The samples were pre-processed into 123-dimensional filterbank features (40 Mel-spaced filterbanks, energy coefficent, 1st and 2nd order delta features, 25ms frames, 10ms frame shift) and normalized to zero-mean and unit variance per sample. The output labels consist of 59 alphabet units (characters, digits etc.) that are obtained with the EESEN pre-processing routines [22]. The task in the CHiME-4 experiments is ASR: sequences of input features are transcribed to sequences of characters. The CER is used as the performance metric.

4.3.2 *Models*

In total, five different models are evaluated: `CONCAT-2CH`, `AVG-2CH`, `STAN-2CH`, `STAN-5CH` and `BEAMFORMIT-5CH`. The two-channel models are trained on channels (2,5), with the low SNR backwards channel 2 and the high SNR front channel 5 [31]. The five-channel models are trained on the five front channels (1,3,4,5,6). Each input channel provides pre-processed filterbank features, and because it was not necessary to use further transformations, the

ID	Model	Train channels	Test channels	# Parameters	et05 real	dt05 real	Attention weights, dt05_real $\bar{\alpha}^1$	$\bar{\alpha}^2$	$\bar{\alpha}^3$	$\bar{\alpha}^4$	$\bar{\alpha}^5$	$\bar{\alpha}^6$
(a)	CONCAT-2CH	2,5	2,5	13.51M	**30.4**	20.4	-	-	-	-	-	-
(a)	AVG-2CH	2,5	2,5	13.15M	36.3	24.6	-	-	-	-	-	-
(a)	STAN-2CH	2,5	2,5	13.17M	**30.4**	**19.9**	-	0.22	-	-	**0.78**	-
(b)	CONCAT-2CH	2,5	5,2	13.51M	57.8	43.1	-	-	-	-	-	-
(b)	AVG-2CH	2,5	5,2	13.15M	36.3	24.6	-	-	-	-	-	-
(b)	STAN-2CH	2,5	5,2	13.17M	**30.4**	**19.9**	-	0.22	-	-	**0.78**	-
(c)	AVG-2CH	2,5	1,3,4,5,6	13.15M	**24.9**	**17.2**	-	-	-	-	-	-
(c)	AVG-2CH	2,5	1,2,3,4,5,6	13.15M	27.0	18.5	-	-	-	-	-	-
(c)	STAN-2CH	2,5	2	13.17M	61.3	47.3	-	**1.00**	-	-	-	-
(c)	STAN-2CH	2,5	5	13.17M	28.8	19.3	-	-	-	-	**1.00**	-
(c)	STAN-2CH	2,5	1,3,4,5,6	13.17M	25.7	17.4	0.19	-	0.18	0.19	**0.23**	0.21
(c)	STAN-2CH	2,5	1,2,3,4,5,6	13.17M	26.4	17.8	0.17	0.07	0.17	0.19	**0.21**	0.19
(d)	STAN-5CH	1,3,4,5,6	1,3,4,5,6	13.17M	26.5	17.7	0.17	-	0.17	0.21	**0.23**	0.22
(d)	BEAMFORMIT-5CH	1,3,4,5,6	1,3,4,5,6	13.15M	**24.2**	**15.9**	-	-	-	-	-	-
(d)	ATTMULTI-E2E [59]	1,3,4,5,6	1,3,4,5,6	∼8M	38.0	26.8	-	-	-	-	-	-
(d)	MASK_NET(ATT) [45]	1,3,4,5,6	1,3,4,5,6	∼18M	26.8	18.2	-	-	-	-	-	-

TABLE 4.2: Results for the CHiME-4 multi-channel ASR experiments. The CER [%] is given for the et05_real and dt05_real subsets. The attention weights for STAN-2CH and STAN-5CH are averaged over all frames of the dt05_real subset. The lowest CER and highest attention weight are printed bold. All models are trained and tested on matched channel configurations, and the CONCAT, AVG and STAN-2CH models are additionally tested on new channel configurations without retraining.

identity transformation function T^i ($t_k^i = f_k^i$) is chosen. The models apply different channel combination strategies: the CONCAT-2CH model concatenates both input channels for classification, and the AVG-2CH averages both input channels before classification. The averaging strategy corresponds to assigning fixed attention weights $\alpha_k^i = 1/2$ to the input frames. The STAN models, depicted in Figure 4.5, compute data-dependent attention weights and implement the channel scoring functions Z^i with 20 LSTM units followed by a single dense unit with a SELU non-linearity (Eq. 4.5), resulting in 11k additional parameters over the AVG model (+0.09% relative). Because the input channels are of the same modality, the same scoring function Z is applied to each channel i, therefore $\theta_{Z^1} = ... = \theta_{Z^N}$. The BEAMFORMIT-5CH model uses a delay-and-sum beamformer [90] which first produces enhanced

waveforms in a separate pre-processing stage so it is not considered as an end-to-end model.

All five models use the same classifier architecture based on 5 layers of bidirectional LSTM cells with 350 units per direction followed by a 59-dimensional output projection, but are individually optimized. The models are trained in an end-to-end fashion for 150 epochs with the ADAM optimizer and the CTC objective. The model with the lowest CER on the development set was used for evaluation.

The evaluation further includes results from related models on multi-channel end-to-end ASR without additional lexicons or language models. The `ATTMULTI-E2E` [59] combines multiple input channels into a single representation with a sensory attention mechanism based on weighted summation. Their attention mechanism shows two main differences to the one used in the `STAN` models: (1) it uses a custom designed neural network cell to compute attention scores while STANs use generic LSTM and dense units and (2) it is not invariant to the re-ordering of input channels, while ours is invariant due to the choice of $\theta_{Z^1} = ... = \theta_{Z^N}$. The `MASK_NET(ATT)` [45] model uses an attention mechanism that selects the reference microphone for a neural beamformer. In contrast to the `ATTMULTI-E2E` and `STAN` models, the channels are not combined by a sensory attention mechanism, but rather by a neural beamformer. The neural beamformer is able to exploit spatial information, which is not considered by `ATTMULTI-E2E` and `STAN`. Both `ATTMULTI-E2E` and `MASK_NET(ATT)` use a CTC+Encoder/Decoder hybrid model that is trained with a joint CTC-attention multi-task objective, while the `STAN` model is trained with an encoder (i.e. the acoustic model) and single CTC objective.

4.3.3 *Results*

The evaluation is carried out on the et05_real and dt05_real subsets and the CER results are reported in Table 4.2. For `STAN-2CH` and `STAN-5CH`, the results include the average attention weight $\left(\bar{\alpha}^i = \frac{1}{K}\sum_{k=1}^{K} \alpha_k^i\right)$ of every input channel obtained on the dt05_real subset ($K = 985619$ frames). Note that the way the attention weights are reported corresponds to the physical CHiME-4 channels, and does not reflect the input channel order.

4.3.3.1 *Two-channel Models & Matched Channel Order*

The models are trained and tested on the channel order (2,5) (see Table 4.2(a)). The `STAN-2CH` and the `CONCAT-2CH` models perform best and achieve similar error rates. `STAN-2CH` shows a relative CER improvement between 16.3% to

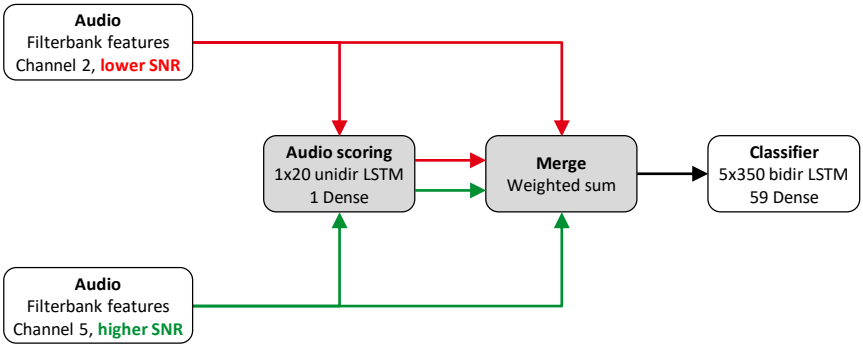


FIGURE 4.5: `STAN` model architecture for two-channel input on the CHiME-4 experiments. The grey boxes correspond to elements that are part of the sensory attention mechanism. The red and green arrows show the Channel 2 and 5 pathways respectively. There is only one network for audio scoring that computes attention scores for all input channels. The scoring functions use relatively fewer units and parameters (11k) than the classifier (13.5M). The `CONCAT` and `AVG` models would replace the sensory attention mechanism by concatenation and averaging operations respectively.

19.1% over `AVG-2CH`. Seemingly, `STAN-2CH` benefits from the automatically learned channel weighting. The average attention weight assigned to channel 5 is 3.5x higher than for the noisy channel 2: $\bar{\alpha}^5 = 3.5 \cdot \bar{\alpha}^2$, and the relation $\alpha_k^5 > \alpha_k^2$ holds true for 94.2% of all $K = 985619$ frames. In other words, by comparing attention weights alone, the higher SNR channel 5 can be identified with 94.2% accuracy.

4.3.3.2 *Two-channel Models & Reversed Channel Order*

The models were originally trained on channel order (2,5) but are then tested on the reversed channel order (5,2) without any re-training (Table 4.2(b)). As expected, the `STAN-2CH` and `AVG-2CH` models show error rates that are identical to the train channel order, as both are invariant to channel order. `CONCAT-2CH` performs worse with the reversed channels and shows a relative CER increase of 90.1% to 111.3% compared to the train channel order.

4.3.3.3 *Channel Addition and Removal*

The `STAN-2CH` and `AVG-2CH` models were originally trained on channels (2,5), but are then tested on new channel configurations without any re-training (Table 4.2(c)). Both models allow the re-use of the same classifier because the merged feature dimensionality does not change with the number of input channels. The `CONCAT-2CH` model is not considered here as it does not allow to re-use of the classifier: the concatenated feature dimensionality grows with the number of input channels, but the classifier expects the same input dimensionality as during training.

Interestingly, both `AVG-2CH` and `STAN-2CH` show improved CER scores when tested with the new channel configurations (1,3,4,5,6) and (1,2,3,4,5,6), without any re-training. Compared to the channel configuration (2,5), `STAN-2CH` shows relative CER improvements between 10.6% to 15.5% and `AVG-2CH` shows relative CER improvements between 24.8% to 31.4%. Both models now achieve similar CER, and the previous advantage of `STAN-2CH` in the two-channel tests is reduced. This is expected, as the benefit of dynamic channel weighting should be smaller when the merged representation mainly (1,2,3,4,5,6) or only (1,3,4,5,6) consists of the five front channels with similar signal quality. The average attention weights computed by `STAN-2CH` seem reasonable with the five front channels at equal levels. When available, the backwards facing channel 2 can be identified by its significantly lower average attention weight $\bar{\alpha}^2$. The operation of the `STAN-2CH` attention mechanism is further illustrated on a sample using channel configuration (1,2,3,4,5,6) and with channel corruption on channels 2 and 3 in Figure 4.6. More examples are shown in the Appendix A.2.

The single-channel tests are only performed for `STAN-2CH`, and keeping only channel 5 results in similar CER compared to the default (2,5) configuration. This is no surprise considering that the sensory attention mechanism already favored channel 5. When keeping only channel 2, the CER becomes significantly worse.

4.3.3.4 *Related Work*

All models in this comparison are trained and tested on the five front channels (Table 4.2(d)). `STAN-5CH` shows error rates that are similar to `MASK_NET(ATT)` [45] and that are significantly lower than `ATTMULTI-E2E` [59].

The `BEAMFORMIT-5CH` model achieves the lowest overall error rates with relative CER improvements of 8.7% to 10.2% over `STAN-5CH` and 5.8% to 8.6% over `STAN-2CH`. While the `BEAMFORMIT-5CH` model shows lower error

rates than the sensory attention mechanism, the underlying beamforming algorithm uses significantly more processing time. In order to generate the enhanced output from the five input channels on a sample of average length 6s, the beamforming algorithm takes 3554ms (CPU) while the attention mechanism of `STAN-5CH` only takes 195ms (CPU) or 25ms (GPU), i.e. 25x to 142x faster (Skylake Xeon CPU with 4.3GHz, GTX 1080 GPU).

4.3.4 *Discussion*

The proposed experiments evaluated STANs on a noisy ASR task with real-world noise. The CHiME-4 dataset provided multi-sensor input from a single audio-modality, and allowed to establish a two-channel scenario where both channels had different SNR levels due to their orientation towards and away from the speaker. This SNR information served as the ground truth to assess the proper function of the STAN attention mechanism. The parameterization of the STAN attention mechanism was kept similar to the previous experiments with synthetic noise (20 LSTM units and 1 dense unit), and added only 11k parameters to a 13.5M parameter network. Despite the increased difficulty in this real-world task, the sensory attention mechanism continued to compute attention weights that correlated with the channel SNR level. In the two-channel scenario, the tuning of the attention weights reduced error rates significantly over the simple averaging of the channels. The dynamic tuning could also adapt to reversed channel orders, while the concatenation model was restricted to its learned channel preference and resulted in higher error rates than the STAN model.

The STAN model was initially trained on a two-channel scenario, but showed remarkable flexibility as it was able to deal with channel addition and removal after training and without any re-training. However, increasing the number of channels showed reduced effectiveness of the channel tuning, as the average model achieved error rates on par with the STAN model for five-channel and six-channel configurations. In the five-channel scenario, the input consisted of the five front channels with similar SNR, eliminating the incentive for SNR-based channel tuning. In the six-channel scenario, the five front channels outnumbered the single backward channel, making averaging already an effective solution. Based on these results, it seems reasonable to assume that STAN are most effective when the number of noisy channels is as high or higher than the number of cleaner channels.

A final evaluation step covered a five-channel scenario and compared the STAN attention mechanism with conventional beamforming. Ultimately, the

comparison shows a trade-off situation between model complexity and error rates. While the beamforming model achieved lower error rates, it also used a separate enhancement stage that is incompatible with the end-to-end paradigm and increases the model complexity over STANs.

Given recent advances in reducing the amount of hand-tuning in feature pre-processing, the next Section 4.4 will address the open question whether the attention mechanism can cope with input features that are less pre-processed than the filterbank features used in this section.

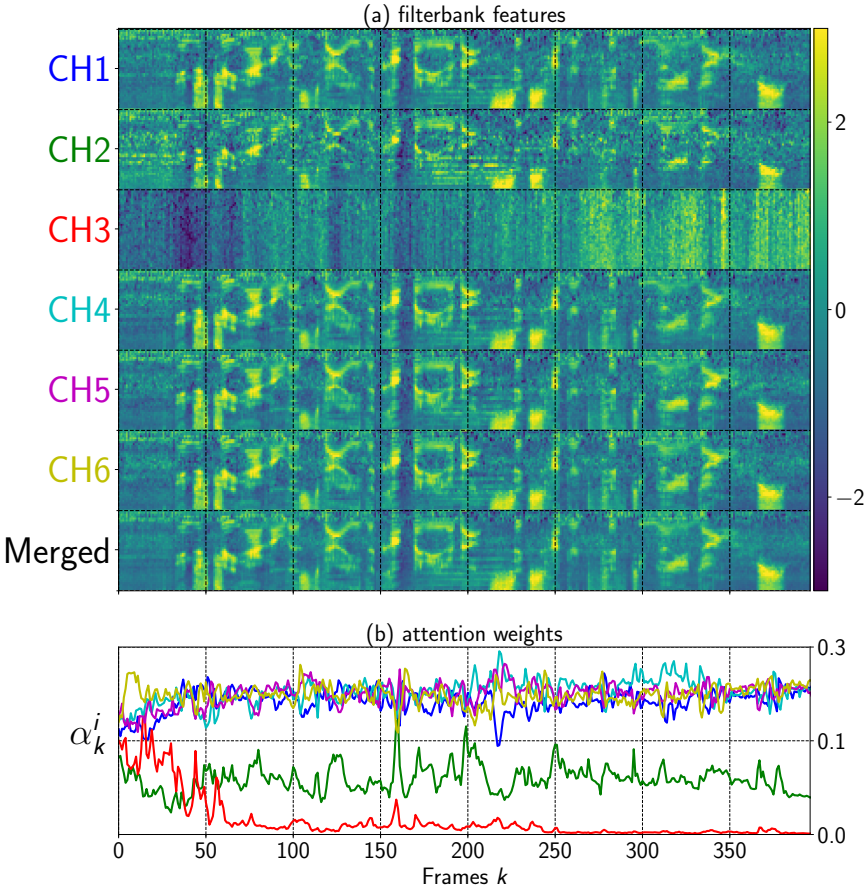


FIGURE 4.6: Operation of `STAN-2CH` on a sample with channel configuration (1,2,3,4,5,6). (a) Filterbank features for the 6 input channels and the merged representation. (b) Attention weights α_k^i for the 6 input channels. The attention weights show three distinct tiers: the cleaner channels (1,4,5,6) are assigned the highest attention weights that are roughly equal for all 4 channels. The weights of noisy channel 2 lie between those of (1,4,5,6) and the highly corrupted channel 3 (isolated case of microphone failure). The merged representation appears to be hardly corrupted by channels 2 and 3.

4.4 MULTI-CHANNEL SPEECH RECOGNITION WITH A CONVOLUTIONAL FRONT-END ARCHITECTURE

The previous Sections 4.2 and 4.3 made fairly conservative choices in terms of audio input features (filterbank features) and end-to-end model (LSTM blocks only). While filterbank features have been a standard choice as input for end-to-end models [22, 59], the spectrogram features have recently found increasing use in end-to-end models, especially in large-scale models [19, 20]. Spectrogram features are obtained with fewer processing steps than filterbank features, as only the log-magnitude spectrogram is required. This reduces the computational burden on the pre-processing stage. Also, various end-to-end models have recently integrated learnable CFEs into the model architecture [20, 44]. In the example of *Deep Speech 2* [20], the CFE improved the noise robustness, and also allowed for a down-sampling in the temporal dimension, reducing the amount of time steps that the ASR systems has to process. In this section, these more recent developments are taken into account, and STANs are evaluated on an end-to-end ASR system with spectrogram input features and an integrated CFE block.

4.4.1 *Neural Network Architecture*

The acoustic model receives as input, the merged representation generated by the STAN sensory attention mechanism detailed in Section 4.1. The end-to-end model is composed of a CFE followed by a stack of LSTM units.

The CFE is made of three convolutional blocks. Each block performs a function f that includes a 2D convolution, a 2D instance normalization [91] and a clipped ReLU non-linearity $\sigma(x) = \mathsf{min}\{\mathsf{max}(x, 0), 20\})$ [92]:

$$f(x) = \sigma(\mathsf{InstanceNorm2d}(\mathsf{Conv2d}(x))) \tag{4.10}$$

The CFE operates on spectrogram features. As the CFE uses a temporal stride of 2 in the first layer, it effectively halves the sequence length and reduces training time. The proposed CFE implementation is closely related to *Deep Speech 2*, where a similar CFE configuration helped to improve error rates especially in noisy conditions [20]. The main difference of the proposed implementation is the use of instance normalization (sample-wise normalization) instead of batch normalization (batch-wise normalization), and the mean and variance statistics from training are not applied during normalization at test time. On the four different noise environments of CHiME-4, using mean and variance statistics computed across samples from

different environments for normalization, decreased the model performance in preliminary experiments, and therefore instance normalization is used.

The CFE is followed by a stack of bidirectional LSTM units and the final output layer is an affine transform to the class labels. The CTC [28] objective is used to automatically learn the mapping and alignment between input features and label sequences. The model is tested with strict end-to-end criteria and without use of external lexicons or language models. The CTC output is decoded in a greedy fashion: at each time step, the most likely label is selected.

4.4.2 *Dataset*

All experiments are carried out as ASR tasks on the CHiME-4 data-set [31] (please refer to Section 4.3.1 for a dataset description). Both real data (tr05_real, 1600 samples) and simulated data (tr05_simu, 7138 samples) are used for training.

The audio samples are pre-processed into 161-dimensional spectrogram features with the short-time Fourier transform (STFT). First, the STFT-coefficients are computed (20 ms frame length, 10ms frame shift, Hamming window) and then the log of the magnitude of the STFT-coefficients is kept. The features are further normalized to zero mean and unit variance per sample. The output labels consist of 59 distinct labels such as characters and digits and are obtained with the EESEN pre-processing routines [22].

4.4.3 *Models*

In total, 5 different models are evaluated: `NOISY`, `BEAMFORMIT`, `MVDR`, `AVG` and `STAN`. All of these models are trained and tested for 10 different random initializations. The `NOISY` model is trained and evaluated only on channel 5. It provides a baseline for a model optimized on the best-performing channel. All other models are trained and tested on the front channels 1/3/4/5/6, but differ in their channel combination strategies. The `BEAMFORMIT` model uses a delay-and-sum beamformer [90], while the `MVDR` model uses a minimum variance distortionless response (MVDR) beamformer based on the implementation provided by the CHiME authors [31]. Both beamformers produce enhanced waveforms in a separate pre-processing stage that is not optimized towards the ASR objective, and so their corresponding models are not considered as end-to-end models. The `STAN` model is depicted in Figure 4.7 and uses the STAN attention mechanism (Section 4.1) to merge

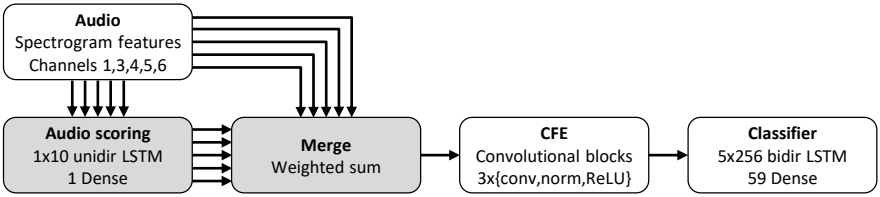


FIGURE 4.7: The `STAN` model architecture for five-channel input on the CHiME-4 experiments. The gray boxes correspond to elements that are part of the sensory attention mechanism. There is only one network for audio scoring that scores all input channels. The scoring functions use relatively fewer units and parameters (7k) than the CFE (1M) or classifier (7.3M). The `AVG` model would replace the sensory attention mechanism by an averaging operation.

the input channels. In order to assess the effectiveness of this attention mechanism, the `STAN` model is compared against an averaging model, `AVG`, that assigns fixed attention weights $\alpha_t^c = 1/5$ for the five input channels. The simple channel concatenation strategy is not included here, because it is not inherently invariant to channel re-ordering (see [59]). Also, results from the previous Section 4.3 showed that concatenation complicates channel addition or removal after training because the acoustic model expects a fixed input dimensionality. The results include the `ATTMULTI-E2E` [59] and `MASK_NET(ATT)` [45] models for comparison, and both models have been described in Section 4.3.

4.4.4 *Training Parameters*

All models are optimized separately, but use the same acoustic model architecture presented in Section 4.4.1: a CFE with 3 layers of convolutional blocks (Table 4.3) followed by 5 layers of bidirectional LSTMs with 256 units in each direction. The final output layer is an affine transform to the 59 output classes. The `STAN` model uses 10 LSTM units followed by a single dense unit with a SELU non-linearity to implement the attention scoring function Z (Equation 4.5), resulting in 7k additional parameters. The models were trained in an end-to-end fashion with the CTC objective [28] and the ADAM optimizer [82] for 150 epochs. The model with the lowest CER on the development set was used for evaluation.

Layer	Channels	Kernel	Stride
L1	32	41x11	2x2
L2	32	21x11	2x1
L3	96	21x11	2x1

TABLE 4.3: 2D convolution filters of the CFE. First dimension is frequency and second dimension is time.

4.4.5 *Results*

The CER obtained on the CHiME-4 development and evaluation sets are reported in Table 4.4. The CER values reflect the average CER and standard deviation collected over 10 runs per model.

4.4.5.1 *Real Noisy Data*

The `BEAMFORMIT` model achieves the lowest overall error rates on both real noisy subsets et05_real and dt05_real, and the `STAN` model is almost on par, with error rates that are relatively by 1.3% to 2.7% higher. Compared to the other models, `STAN` shows a relative CER improvement of 0.4% to 1.9% over the `AVG` model and 19.0% to 20.8% over `NOISY`. The `MVDR` model shows better results than the single channel `NOISY` model, but is not competitive with the other approaches on real noisy data.

Results from related work report higher error rates. The `STAN` model shows a relative CER improvement of 12.7% to 16.5% over `MASK_NET(ATT)` and 38.4% to 43.3% over `ATTMULTI-E2E`. The higher error rates of `MASK_NET(ATT)` and `ATTMULTI-E2E` may originate from their hybrid CTC+Encoder/Decoder acoustic model unlike our simple CTC model. The number of parameters of the `STAN` model (8.031M) also compares favorably against those of `ATTMULTI-E2E` (∼8M) and `MASK_NET(ATT)` (∼18M). Note that the latter implements the neural beamformer part with an estimated ∼10M parameters, while the sensory attention mechanism only uses 7k parameters.

4.4.5.2 *Simulated Noisy Data*

The `MVDR` model clearly achieves the lowest CER on both simulated noisy subsets et05_simu and dt05_simu and yields significantly lower error rates than it did on the real noisy data. For MVDR beamforming, better performance on simulated data was also reported by the CHiME-4 authors and explained with

the absence of reverberation in the simulated data [31]. The `MASK_NET(ATT)` model performs significantly better than `STAN` on dt05_simu, but worse on et05_simu. The `BEAMFORMIT` model performed worse than the single channel `NOISY` model on et05_simu. This result may be explained by the separate optimization of the beamforming and acoustic model components. A hearing inspection also led to the hypothesis that the simulated noisy data itself could explain the unexpected findings: at times, the simulation process introduces residual speech artifacts on channels 1/3/4/6 but produces a cleaner channel 5 signal[3].

4.4.5.3 *New Channel Configurations*

The flexibility and interpretability of the sensory attention mechanism is demonstrated through additional experiments on dt05_real. The CER of the `STAN` and `AVG` models are tested for the cases of channel re-ordering, channel addition and channel removal. The models are not re-trained for these new channel configurations. The CER results are reported in Table 4.5 along with the average attention weight $\left(\bar{\alpha}^c = \frac{1}{T}\sum_{t=1}^{T} \alpha_t^c\right)$ of every channel c of `STAN`, computed over all $T = 989608$ frames of dt05_real. The results are collected from 10 runs per model. Note that the way that the attention weights are reported corresponds to the physical tablet channels, and does not reflect the channel order. The `AVG` model assigns equal attention weights to all N channels, i.e. $\alpha_t^c = \frac{1}{N}$.

As expected, both models are invariant to channel re-ordering and yield identical CER for channel orders 6/5/4/3/1 and 1/3/4/5/6. Adding the noisy channel 2 (1/2/3/4/5/6) leads to a smaller increase in CER for `STAN`. In fact, `STAN` suppresses channel 2 as seen by the lower attention weight α_t^2 of this channel when compared to the other channels. This indicates a good generalization of the sensory attention mechanism because it was not trained on the data from channel 2. For all channel configurations, channel 2 has the lowest attention weight and channel 5 has the highest attention weight whenever either one is present. When removing channels, `STAN` has an increased advantage and shows a relative CER improvement of up to 11.9% over `AVG` in the channel configuration 2/5. For this configuration, the results show that the attention mechanism is quite accurate: $\alpha_t^5 > \alpha_t^2$ holds true for 96.2% of all frames. In other words: by comparing attention weights alone, the SNR channel 5 can be identified with 96.2% accuracy. The high interpretability of the attention weights is further confirmed by the plots of

3 e.g. sample 'M06_447C0216_STR' from et05_simu

the input features and attention weights for the channel configuration 2/5 in Figure 4.8.

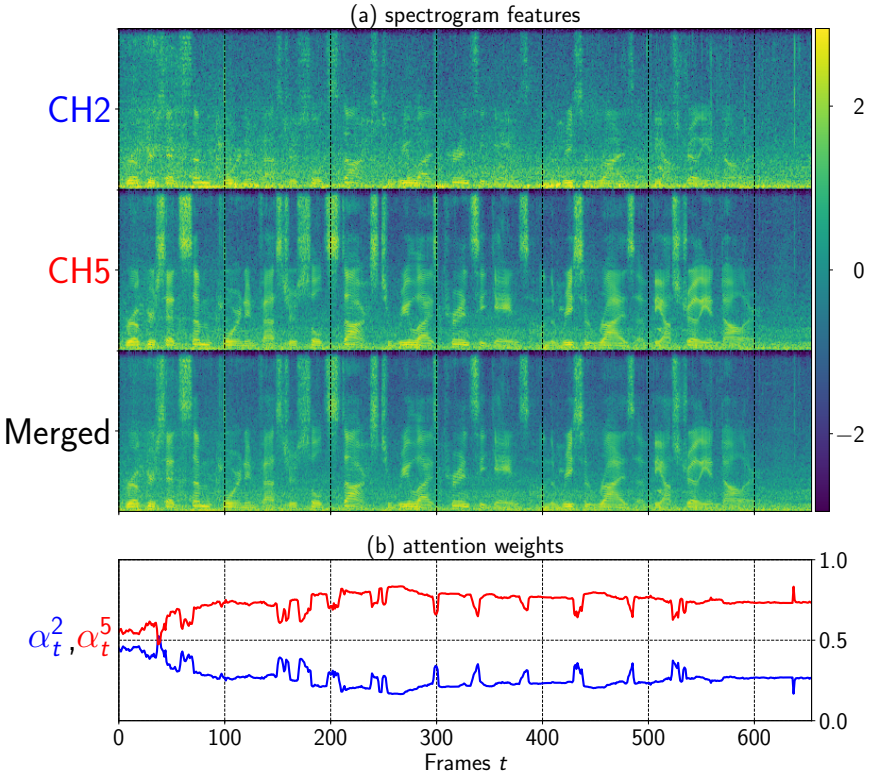


FIGURE 4.8: Operation of `STAN` on a sample with channel configuration 2/5. (a) Spectrogram features for the two input channels and the merged representation. (b) Attention weights for the two input channels. The merged representation is dominated by channel 5, as evident by the higher attention weights of this channel which has less noise.

4.4.6 *Discussion*

The STAN end-to-end model was evaluated in the context of multi-channel ASR and an acoustic model architecture similar to Deep Speech 2. Instead of the filterbank features that were used in previous experiments, this model operates on spectrogram features that require less pre-processing. The network

TABLE 4.4: CER [%] results on CHiME-4 ASR experiments. The CER is given as the average ± standard deviation over 10 runs. The best results are printed in bold. For models from related work, only one run is available, and the parameter counts were estimated.

Model	Parameters	dt05_real	dt05_simu	et05_real	et05_simu
`NOISY`	8.024M	19.2 ± 0.4	19.2 ± 0.4	28.9 ± 0.4	24.6 ± 0.5
`AVG`	8.024M	15.5 ± 0.6	17.8 ± 0.6	23.5 ± 0.6	23.7 ± 0.7
`STAN`	8.031M	15.2 ± 0.1	17.2 ± 0.2	23.4 ± 0.2	22.4 ± 0.4
`BEAMFORMIT`	8.024M	**14.8 ± 0.3**	17.3 ± 0.3	**23.1** ± 0.3	26.0 ± 0.4
`MVDR`	8.024M	17.9 ± 0.2	**12.5 ± 0.2**	28.0 ± 0.5	**15.8** ± 0.3
`ATTMULTI-E2E` [59]	∼8M	26.8	26.5	38.0	32.9
`MASK_NET(ATT)` [45]	∼18M	18.2	15.3	26.8	23.7

TABLE 4.5: CER [%] results on the dt05_real subset of CHiME-4 for new channel configurations. The CER is given as the average ± standard deviation over 10 runs. The attention weights for `STAN` are first averaged over all frames of dt05_real, and then averaged over all 10 runs. The standard deviations for the attention weights are not given for space constraints, but the maximum standard deviation seen is 0.017. The lowest CER and highest attention weight are printed in bold.

	CER [%]		`STAN` attention weights					
Channels	`AVG`	`STAN`	$\bar{\alpha}^1$	$\bar{\alpha}^2$	$\bar{\alpha}^3$	$\bar{\alpha}^4$	$\bar{\alpha}^5$	$\bar{\alpha}^6$
1/3/4/5/6	15.5 ± 0.6	**15.2 ± 0.1**	0.19	-	0.18	0.22	**0.23**	0.18
6/5/4/3/1	15.5 ± 0.6	**15.2 ± 0.1**	0.19	-	0.18	0.22	**0.23**	0.18
1/2/3/4/5/6	16.6 ± 0.6	**16.0 ± 0.1**	0.17	0.12	0.16	0.19	**0.21**	0.16
2/3/4/5	17.9 ± 0.6	**16.8 ± 0.1**	-	0.18	0.23	0.28	**0.30**	-
2/3/5	19.4 ± 0.6	**17.8 ± 0.2**	-	0.26	0.32	-	**0.42**	-
2/5	22.6 ± 0.9	**19.9 ± 0.2**	-	0.38	-	-	**0.62**	-
2	45.9 ± 0.3	**45.8 ± 0.3**	-	**1.00**	-	-	-	-
5	17.6 ± 0.5	**17.6 ± 0.2**	-	-	-	-	**1.00**	-

also embeds additional convolutional layers to down-sample and process the input. Both architectural changes are especially interesting for online ASR, where the reduction of the computational load is welcome to save hardware resources and compute time.

In a five-channel scenario and compared to the filterbank STAN model from the previous Section 4.3, the spectrogram STAN model reduced CER by relatively 11.7% to 16.4% on real noisy data. The results showed that the STAN attention mechanism still managed to produce interpretable attention weights on spectrogram features. Despite the fact that the attention weights correlated well with the SNR level, the STAN model achieved error rates almost on par and only slightly better than the averaging model. These findings confirms previous results from the five-channel and six-channel experiments in Section 4.3. However, the sensor removal experiments also show input configurations where STANs are more effective. The two-channel experiment uses a higher SNR front channel and a lower SNR back channel, and the STAN model achieves 11.9% relatively lower CER than the averaging model. Notably, STANs were not trained on the noisy channel 2, but the attention mechanism still managed to generalize.

On a final note, the results further confirm the importance of multi-channel processing to increase the noise robustness of end-to-end models. The multi-channel STAN model achieved a 20.0% to 20.9% relatively lower CER than a single channel model. Similar improvements are seen with the model using the Beamformit algorithm, but the underlying beamforming algorithm is not compatible with the end-to-end paradigm. The MVDR algorithm for beamforming only achieved good results for the simulated noisy data, were the speaker position is fixed. For real-world noisy data, MVDR was not competitive with either STANs or the model using the Beamformit algorithm.

4.5 CONCLUSION

This thesis chapter presented the STAN end-to-end model that embeds a sensory attention mechanism for multi-sensor setups. The STAN model is trainable in an end-to-end fashion and compatible with uni-modal and multi-modal multi-sensor input. The attention mechanism is built with default neural network units and increases the total amount of parameters by only a small amount. By processing sensor input only, the attention mechanism dynamically tunes its attention towards higher SNR sensors. Experiments on multi-modal AVSR and multi-channel ASR showed that STANs achieve lower error rates over single sensor configurations. Across all experiments, the sensory attention mechanism of STANs also performed on par or better than averaging or concatenation strategies for multi-sensor combination, with the additional benefit of computing highly interpretable attention weights. Notably, the sensory attention mechanism was able to robustly process different modalities and feature types, i.e. video features and different audio features such as filterbank and spectrogram features. Given the small increase in neural network parameters and competitive performance in all experiments, the sensory attention mechanism provides an effective solution to leverage multi-sensor input for increased noise robustness in end-to-end models.

However, the sensory attention mechanism may be further optimized. In fact, it was formulated in a modality-independent way to guarantee compatibility with arbitrary input modalities. The compatibility comes at a cost, and for uni-modal scenarios such as multi-channel ASR, there may be modality-specific optimizations to exploit. In particular, future work could include spatial information in the attention mechanism, as is also used by most beamforming algorithms for better multi-channel enhancement.

5

PARAMETER UNCERTAINTY FOR END-TO-END MODELS

In the previous two chapters, the noise robustness of end-to-end models has been increased by training the acoustic model with a curriculum learning strategy (Chapter 3) and by leveraging multi-sensor input with sensory attention (Chapter 4). The curriculum learning strategy did not make any changes to the underlying neural network model, and the sensory attention mechanism introduced only a small number of additional neural network units at the input stage - other than that, the acoustic model was left unchanged. In this chapter, the acoustic model architecture is more radically changed by using *parameter uncertainty* throughout all of the acoustic model weights. The proposed architectural change improves the networks performance in domain adaptation for improved noise robustness, but also acts as a regularizer and a helpful tool for parameter pruning.

The text in this chapter is adapted from published work in [93] and the remainder of this chapter is organized as follows. The neural network with parameter uncertainty through probabilistic parameters is presented in Section 5.1, and the baseline experiments are described in Section 5.2. The probabilistic network is further evaluated for parameter pruning in Section 5.3 and domain adaptation in Section 5.4. A final conclusion is given in Section 5.5.

5.1 PROBABILISTIC END-TO-END MODELS

5.1.1 *Random Variable Parameters*

We consider end-to-end models for ASR that transcribe speech input to text output with a single neural network. A conventional end-to-end model consists of a set of $n = 1, ..., N$ *deterministic* parameters $\theta = \{\theta_1, ..., \theta_N\}$ that represent the weights and biases of the neural network units. In this work, we consider *probabilistic* parameters $\Theta = \{\Theta_1, ..., \Theta_N\}$ for the model. While there are many possible ways to define the random variables Θ_n, we choose a Gaussian distribution such that $\Theta \sim \mathcal{N}(\mu, \sigma)$. Note that every parameter Θ_n is described with a parameter-specific mean μ_n and standard deviation σ_n.

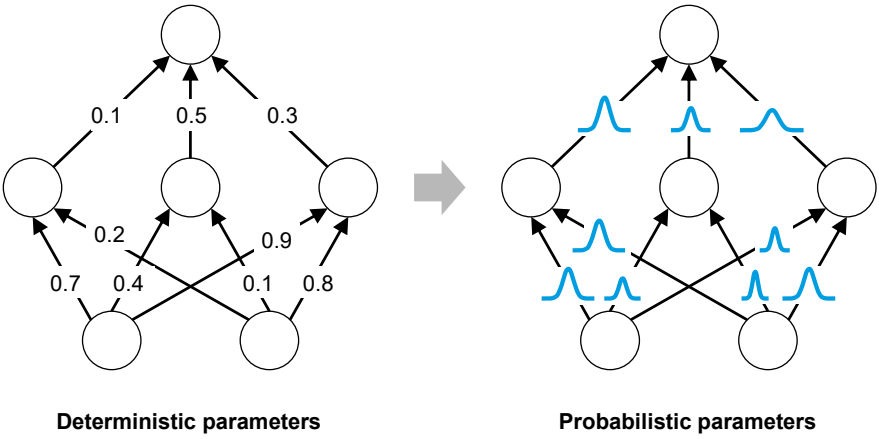


FIGURE 5.1: An illustration of network connections for deterministic (left) and probabilistic parameters (right). When using deterministic parameters, each connection value is represented by a real value. When using probabilistic parameters, each connection value is represented by a probability distribution (e.g. a Gaussian distribution as used in this work.)

The mean and standard deviation represent the expected parameter value and its uncertainty. When using probabilistic neural networks, the network connections are modeled by probability distributions instead of real values as depicted in Figure 5.1.

Similar to related work [66–68], we define a parameter-specific signal-to-noise ratio:

$$\mathrm{SNR}_n = \frac{|\mu_n|}{\sigma_n} \tag{5.1}$$

Note that a parameter with $\mathrm{SNR}_n \to \infty$ could be interpreted as a deterministic parameter. The results of previous studies [66–68] imply that SNR levels can be used to identify the important parameters that are useful for solving a task. The SNR level can be computed for a specific parameter on a *local* level, or for the whole network on a *global* level, e.g. by computing the average or median SNR level of all parameters.

5.1.2 *Training with Random Variables*

The training process optimizes both the mean $\mu \in \mathbb{R}^N$ and standard deviation $\sigma \in \mathbb{R}^N$ of the $n = 1, ..., N$ network parameters. The standard deviation σ is parameterised with the proxy parameter $\beta \in \mathbb{R}^N$ and the softplus function [94] $\sigma = \log(\exp(\beta) + 1)$ to ensure that σ is positive (see Figure 5.2). A training procedure similar to [66] is used that updates network parameters as described in Eq. (5.2)-(5.5):

$$\epsilon \sim \mathcal{N}(0, 1) \tag{5.2}$$

$$\theta = \mu + \log(\exp(\beta) + 1) \cdot \epsilon \tag{5.3}$$

$$\mathcal{L} = f(\theta, x, y) \tag{5.4}$$

$$\mu', \beta' \leftarrow \text{optimizer}(\mu, \beta, \nabla\mathcal{L}_\mu, \nabla\mathcal{L}_\beta) \tag{5.5}$$

The following procedure is repeated for every mini batch: First, noise samples $\epsilon \in \mathbb{R}^N$ are drawn from a standard normal distribution (Eq. (5.2)). The noise samples are scaled by σ and shifted by μ to compute the parameters $\theta \in \mathbb{R}^N$ (Eq. (5.3)). For the forward pass, the parameters θ and the network input x and target labels y are used to compute the loss $\mathcal{L}$ (Eq. (5.4)). For the backward pass, the gradients $\nabla\mathcal{L}_\mu, \nabla\mathcal{L}_\beta$ are computed and the parameters μ, β are updated with an optimizer (Eq. (5.5)).

The first two steps described in Eq. (5.2) and (5.3) are referred to as the *reparameterization trick* in the literature [95] and yield the same effect as sampling θ from $\mathcal{N}(\mu, \sigma(\beta))$, but they keep the sampling operation differentiable wrt. μ, β. Training with probabilistic parameters is not different from training with deterministic parameters with respect to the loss function (Eq. (5.4)) and the optimizer (Eq. (5.5)). Note that the loss function and the optimizer can be of arbitrary choice.

5.1.3 *Testing with Random Variables*

During inference, probabilistic parameters provide a choice to either use an expected parameter value or to sample parameters. In this work, the inference forward pass is carried out with the mean parameter μ to produce the network output o as described in Eq. (5.6) and (5.7):

$$\theta = \mu \tag{5.6}$$

$$o = f(\mu, x) \tag{5.7}$$

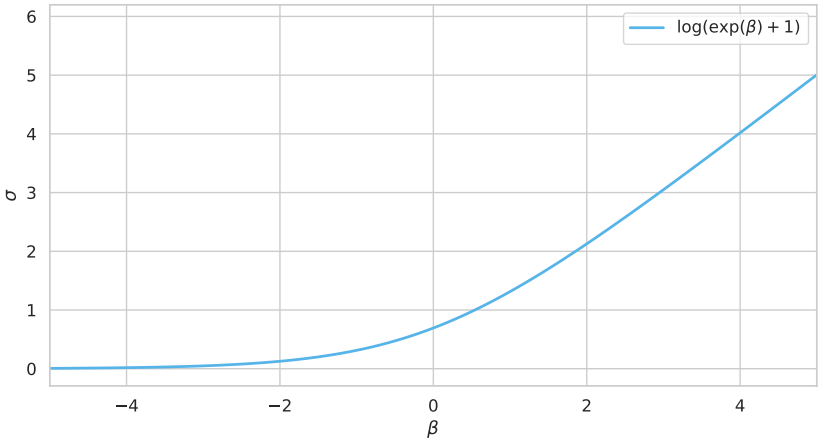


FIGURE 5.2: The softplus function maps the neural network parameter $\beta \in \mathbb{R}$ to the positive-only standard deviation $\sigma \in \mathbb{R}^+$.

Related work has explored parameter sampling during inference [66, 67] to generate model ensembles for improved accuracy.

5.1.4 *Deterministic vs. Probabilistic Parameters*

The proposed probabilistic parameters render the model architecture and training process more complex. First, there is a 2x increase in the number of parameters because the Gaussian distribution that defines each network connection requires 2 parameters in the form of a mean and a standard deviation. Second, during training time, there is an additional sampling step to generate the parameter values. However, the additional complexity brings the benefit of using the SNR level as a proxy measure for parameter importance. Notably, the cost function remains unchanged when using the probabilistic parameters. This is an important feature that makes the proposed probabilistic networks compatible with any end-to-end objective for ASR, and also with other tasks than ASR.

5.1.5 *Related Work*

Probabilistic neural network models have been mostly explored from a variational inference perspective [66–68, 96]. The embedding in a Bayesian

framework allows these approaches to introduce additional loss terms that are interpreted as parameter complexity cost terms [66, 67] or minimum description length cost terms [68, 96]. In contrast, this work develops probabilistic networks from a parameter-based perspective, and there are no additional loss terms that could be developed by a Bayesian interpretation of the network.

To the best of the authors knowledge, only one study evaluated probabilistic networks for end-to-end ASR [68]. This study used a LSTM-based CTC model with 140k parameters on the 5h TIMIT dataset. The effect of parameter pruning was evaluated for a single probabilistic network, and no adaptation scenario was considered. In contrast, this study evaluates multiple probabilistic networks with different SNR levels for both pruning and domain adaptation scenarios. Even the non-ASR studies investigated only SNR-based pruning, but no continual learning scenarios such as domain adaptation.

Probabilistic networks with Gaussian parameters are closely related to the weight noise regularizer for recurrent neural networks (RNNs) [97]. This regularizer adds noise from a normal distribution $\mathcal{N}(0, \sigma_r)$ to the network parameters before the forward pass is carried out in training. The standard deviation σ_r is a single scalar hyperparameter used for all network parameters and is not updated during training [21, 98]. In contrast, this study uses a separate standard deviation per parameter and performs gradient-based updates on the standard deviation in training.

5.2 BASELINE EXPERIMENTS

5.2.1 *Datasets*

All experiments are carried out as ASR tasks on the WSJ [76] and CHiME-4 [31] datasets presented in Table 5.1. The WSJ dataset provides single channel read speech data recorded in *clean* conditions. The CHiME-4 dataset provides read speech data recorded from a 6-channel tablet in *noisy* conditions (bus, street junction, cafe and pedestrian area). In this work, only the channel 5 data is used that leads to the lowest error rates on CHiME-4.

The audio data was pre-processed into 123-dimensional filterbank features (25ms frames, 10ms frame shift, 40 Mel-spaced filterbanks, energy coefficient, 1st and 2nd order delta) and normalized to zero-mean and unit-variance per sample. Both the WSJ and CHiME-4 datasets use the same alphabet of 59 tokens (characters, digits etc.) as output labels which were obtained with the EESEN pre-processing routines [22]. The CER is used as the performance metric.

Dataset	Subset	# hours	Speech domain	Comment
WSJ	train_si284	81.0	clean	-
WSJ	test_dev93	1.0	clean	-
WSJ	test_eval92	0.7	clean	-
CHiME-4	tr05_simu_real	18.0	noisy	only CH5
CHiME-4	dt05_real	2.5	noisy	only CH5
CHiME-4	et05_real	2.0	noisy	only CH5

TABLE 5.1: Datasets used for experimentation. © 2019 IEEE

5.2.2 *Model Architecture*

All models share the same basic architecture depicted in Figure 5.3: 5 layers of bidirectional LSTMs [80] with 320 units in each direction and a final 640x59 projection to the output labels. The deterministic models use default LSTM units and consist of a parameter set θ_D with LSTM weights w^{LSTM}, biases b^{LSTM} and projection weights w^{PROJ} (Eq. (5.8), ∼11M parameters).

$$\theta_D = \{w^{LSTM}, b^{LSTM}, w^{PROJ}\} \tag{5.8}$$

The probabilistic models use LSTMs with Gaussian weights and consist of a parameter set θ_P with LSTM weight means μ^{LSTM}, parameterised weight standard deviations β^{LSTM}, biases b^{LSTM} and projection weights w^{PROJ} (Eq. (5.9), ∼22M parameters).

$$\theta_P = \{\mu^{LSTM}, \beta^{LSTM}, b^{LSTM}, w^{PROJ}\} \tag{5.9}$$

The parameters w^{LSTM}, μ^{LSTM} and w^{PROJ} are initialized with the Xavier uniform initialization [81] according to Eq. (5.10). The same random seed is used to ensure that probabilistic and deterministic models start training with identical weight and weight mean, i.e. $w^{LSTM} = \mu^{LSTM}$. All biases b^{LSTM} are initialized to 0. The parameterized standard deviation β^{LSTM} is initialized according to Eq. (5.11). This initialization results in an average SNR of 1.0 for the LSTM weight parameters, and gives the same convergence speed during training as a network that uses the deterministic parameters. The convergence characteristics of different SNR level initializations on the WSJ dataset is shown in Figure 5.4. Networks with lower SNR initializations take longer to converge due to their high noise level, and may even fail to converge when the noise level of the initialization is too high.

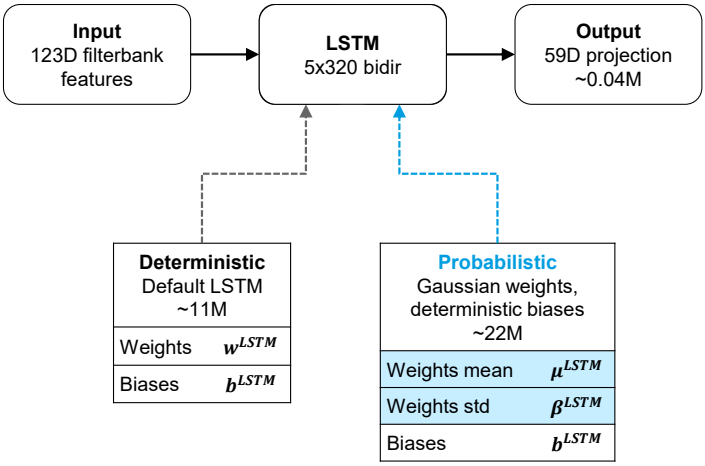


FIGURE 5.3: The model architecture used for experimentation. The deterministic model uses default LSTM units and consists of $\sim$ 11M parameters. The probabilistic model uses LSTM units with Gaussian weights and deterministic biases and consists of $\sim$ 22M parameters.

$$A_{ij} \sim \mathcal{U}\left(-\frac{\sqrt{6}}{i+j}, \frac{\sqrt{6}}{i+j}\right), A \in \mathbb{R}^{i \times j} \tag{5.10}$$

$$B_{ij} = \log\left(\exp\left(\frac{1}{2}\frac{\sqrt{6}}{i+j}\right) - 1\right), B \in \mathbb{R}^{i \times j} \tag{5.11}$$

5.2.3 *Training*

All of the followingly presented models are trained for 25 different random initializations. The models differ in training set, parameter type and loss function. An overview is given in Table 5.2. The baseline deterministic models `D/WSJ`, `D/CHiME` and `D/MIX` use deterministic LSTM units (Eq. (5.8)); and are trained on train_si284, tr05_simu_real and the combined train_si284 + tr05_simu_real subsets respectively. The probabilistic models `P/WSJ/`λ_β use LSTMs with Gaussian weights (Eq. (5.9)) and are trained on train_si284. All models are trained with the CTC loss function $\mathcal{L}_{CTC}$ [28] and the Adam optimizer (learning rate 1e-3) [82] for 25 epochs, and the model from the epoch with the lowest CER on test_dev93 is selected for evaluation. For the

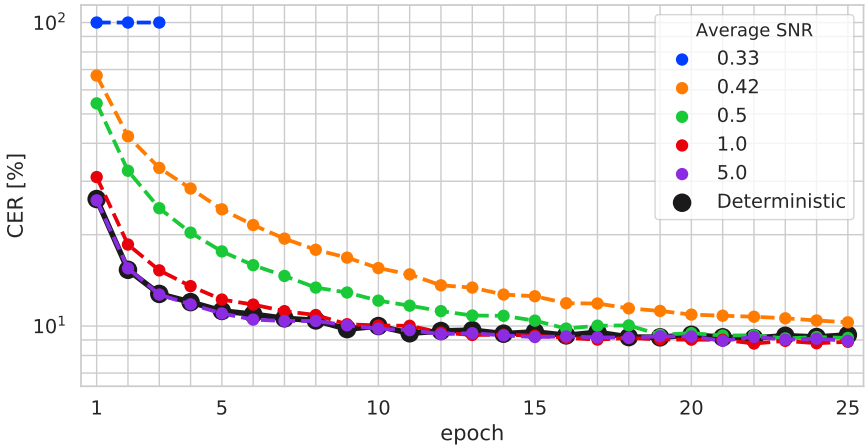


FIGURE 5.4: CER on the WSJ validation set test_dev93 during training on train_si284. The deterministic model (black) converges fastest. The probabilistic models (colored) converge faster when the average SNR at initialization is higher. The probabilistic model with average SNR of 0.33 did not converge, so training was stopped.

probabilistic models, lower SNR parameters are enforced by using weight decay $\mathcal{L}_\beta = ||\beta||_2^2$ on the parameterized weight standard deviation β^{LSTM}. The decay term L_β is scaled by the factor λ_β such that the complete loss function is $\mathcal{L} = \mathcal{L}_{CTC} + \lambda_\beta \mathcal{L}_\beta$. The hyperparameter λ_β is tuned by a grid search in the range 5e-7 to 1e-5.

All neural network models have been implemented in PyTorch [99]. The default LSTM units make use of the fastest publicly available LSTM implementation (cuDNN kernel [100]). The Gaussian weight LSTM units use a modified version of the cuDNN kernel wrapper that implements an additional parameter sampling step as defined in Eq. (5.2). Despite the additional sampling, the models with Gaussian weight LSTMs train only 4% slower than the models with default LSTMs[1].

5.2.4 *Testing*

All models are tested with strict end-to-end criteria and without the use of external language models. The CTC output is decoded in a greedy fashion:

1 measured with a GTX 1080 GPU during WSJ training, default 5x320 bidir LSTM model.

Model	Type	Training set	Cost function	λ_β range
P/WSJ/λ_β	Probabilistic	train_si284	$\mathcal{L}_{CTC} + \lambda_\beta \mathcal{L}_\beta$	0 ... 1e-5
D/WSJ	Deterministic	train_si284	$\mathcal{L}_{CTC}$	-
D/CHiME	Deterministic	tr05_simu_real	$\mathcal{L}_{CTC}$	-
D/MIX	Deterministic	train_si284, tr05_simu_real	$\mathcal{L}_{CTC}$	-

TABLE 5.2: The models used for baseline training.

at every time step, the label with the highest probability is selected. The probabilistic models are tested with the mean weights μ^{LSTM}, i.e. the LSTM parameters are not sampled during testing.

5.2.5 *Results for Probabilistic Models*

As a first evaluation step, the impact of the loss scaling term λ_β on the CER performance of the probabilistic models P/WSJ/λ_β is analyzed. The term λ_β controls the strength of the regularizer $\mathcal{L}_\beta$ that increases the standard deviation of the probabilistic parameters. After the training is completed for 25 runs, the CER results of the models D/WSJ and P/WSJ/λ_β are reported in Figure 5.5. The loss scaling term λ_β is varied from 0 to 1e-5 and the CER is measured on the WSJ test_eval92 subset. For $0 \leq \lambda_\beta \leq 2e-6$, the probabilistic models P/WSJ/λ_β achieve 3.1% to 4.6% relatively lower median CER than the deterministic model D/WSJ. For $\lambda_\beta > 2e-6$, the CER of the probabilistic models increases monotonically from 6.5% CER ($\lambda_\beta = 2e-6$) up to 10.3% CER ($\lambda_\beta = 5e-5$). The results indicate that the loss scaling term λ_β has a strong influence on the CER performance.

In a second evaluation step, the SNR statistics of the Gaussian LSTM weights are analyzed. After the training is completed for 25 runs, the SNR statistics are computed. The median SNR level of the probabilistic parameters is plotted as a function of the loss scaling term λ_β in Figure 5.6. The median SNR of P/WSJ/λ_β is decreasing monotonically from 3.0 ($\lambda_\beta = 0$) to 0.6 ($\lambda_\beta = 1e-5$). In other words, with increasing λ_β, the network gets more noisy. This indicates that the SNR level is indeed controllable by the additional cost term $\mathcal{L}_\beta$ on the parameterized standard deviation β^{LSTM}. Also, the strong increase of the CER results for $\lambda_\beta > 2e-6$ may now be explained by the low SNR level of these networks.

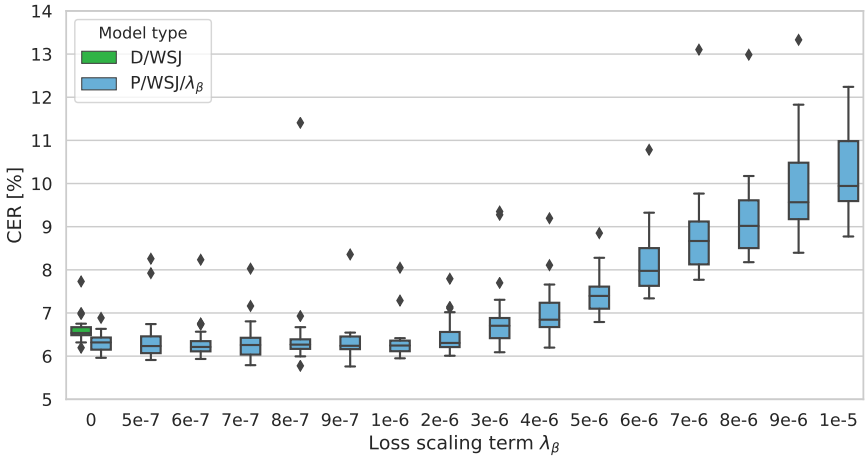


FIGURE 5.5: CER [%] results for testing on WSJ test_dev93 when varying the loss scaling term λ_β. For $0 \leq \lambda_\beta \leq 2e-6$, the probabilistic model `P/WSJ/`λ_β outperforms the deterministic model `D/WSJ` by a small margin. For $\lambda_\beta > 2e-6$, the CER is increasing, and also the range of possible CER results is increasing.

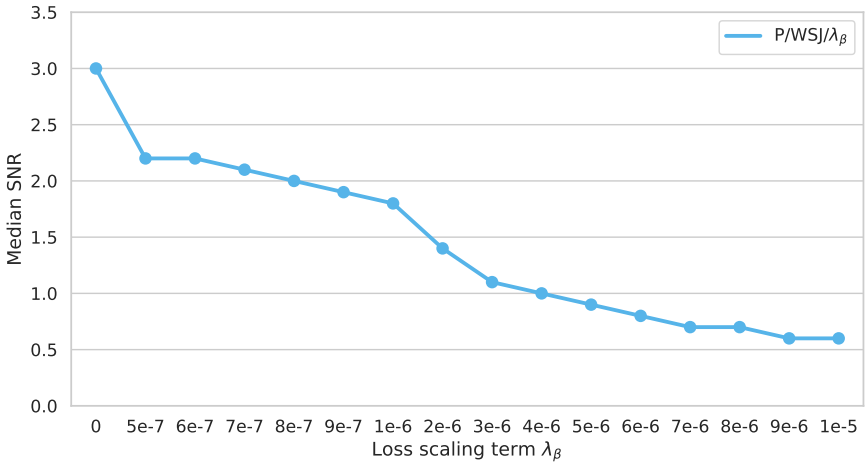


FIGURE 5.6: Median SNR of the Gaussian LSTM weights as a function of the loss scaling term λ_β, collected from 25 runs. With increasing λ_β, the probabilistic network weights decrease in SNR. In other words, the network gets noisier when λ_β increases.

Model	Median	Mean±Std	Min	Max
P/WSJ/0	3.0	3.7±3.0	6.1e-9	123.8
P/WSJ/1e-6	1.8	2.3±1.9	1.6e-8	73.6
P/WSJ/4e-6	1.0	1.2±1.1	3.8e-9	61.4

TABLE 5.3: SNR statistics for the LSTM weights of the probabilistic models obtained after a completed training on train_si284. The statistics are computed over 25 runs following the definition $\text{SNR} = \frac{|\mu^{LSTM}|}{\sigma^{LSTM}}$.

For further evaluation, three probabilistic models with λ_β =0, 1e-6 and 4e-6 are selected. The model P/WSJ/0 is the baseline probabilistic model where only the CTC loss $\mathcal{L}_{CTC}$ isused. The model P/WSJ/1e-6 achieves the lowest CER of all models and the model P/WSJ/4e-6 achieves CER similar to D/WSJ, but at a low SNR. The final selection of models with λ_β =0, 1e-6 and 4e-6 covers median SNR levels of 3.0, 1.8 and 1.0 as reported in Table 5.2. Interestingly, despite median SNR levels of 1.0 to 3.0, the SNR range is much larger and goes from around 0 up to 61.4 (P/WSJ/4e-6), 73.6 (P/WSJ/1e-6) and 123.8 (P/WSJ/0). Plotting the cumulative distribution function (CDF) of the parameter SNR levels in Figure 5.7 reveals that in any model, parameters at SNR>50 are rather rare.

A final SNR analysis shows the SNR level over LSTM layers in Figure 5.8. While P/WSJ/0 shows a monotonically increasing SNR levels from 2.5 to 3.5 between the first and the last network layer, both P/WSJ/1e-6 and P/WSJ/4e-6 maintain a stable SNR level close to the respective median SNR throughout all layers.

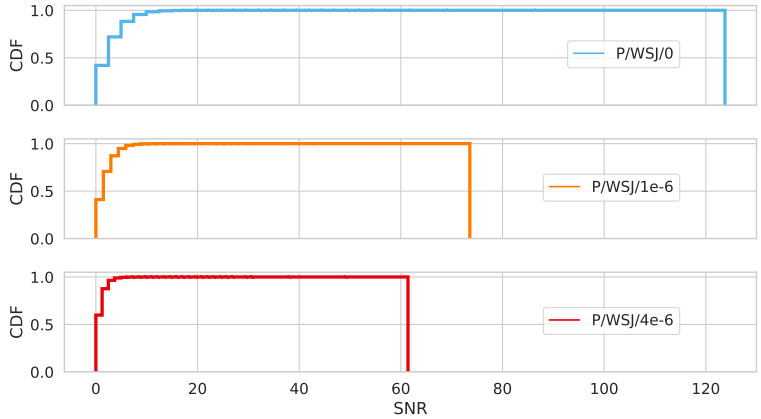


FIGURE 5.7: The cumulative distribution function (CDF) of the parameter SNR levels for the models `P/WSJ/0`, `P/WSJ/1e-6` and `P/WSJ/4e-6`. The range of SNR levels is decreasing when λ_β increases. Even though parameters with SNR > 50 do exist in any model, they are rare compared to lower SNR parameters.

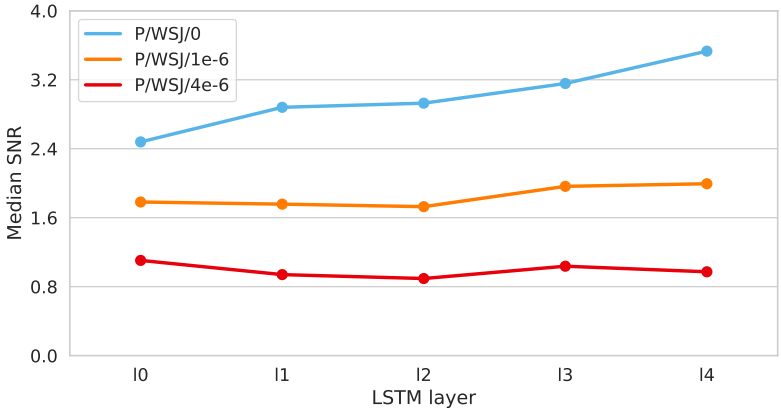


FIGURE 5.8: The median SNR level as a function of the LSTM layer in the network, collected over 25 runs. The default probabilistic model `P/WSJ/0` shows a monotonically increasing median SNR from the first to the last layer. The noisier models `P/WSJ/1e-6` and `P/WSJ/4e-6` stay close to the median SNR level throughout all LSTM layers.

	WSJ		CHiME-4	
Model	test_eval92	test_dev93	et05_real	dt05_real
`D/MIX`	**6.3±0.2**	**8.6±0.2**	**33.8±0.4**	**21.8±0.3**
`D/CHiME`	18.0±0.4	22.0±0.3	37.3±0.4	26.7±0.3
`D/WSJ`	6.6±0.3	9.2±0.4	57.1±0.6	44.7±0.6
`P/WSJ/0`	**6.3±0.2**	8.8±0.2	56.4±0.6	43.9±0.7
`P/WSJ/1e-6`	**6.3±0.4**	8.7±0.5	56.0±0.5	43.5±0.6
`P/WSJ/4e-6`	7.0±0.6	9.5±0.7	56.2±0.5	44.0±0.7
`Single-E2E` [58]	-	-	40.9	29.5
`ESPnet` [44]	7.6	10.1	-	-

TABLE 5.4: Baseline CER [%] results for clean speech (WSJ) and noisy speech (CHiME-4). The CER is given as average ± standard deviation over 25 runs. For the models `Single-E2E` and `ESPnet`, there is only one run available. The lowest CER on each subset is printed bold.

5.2.6 *Results for Domain-mismatch Conditions*

The baseline evaluation results on the WSJ and CHiME-4 test and development sets are reported in Table 5.4. All results are given after the training is completed for 25 runs. The deterministic model `D/MIX` trained on both clean speech (WSJ) and noisy speech (CHiME-4) achieves the lowest CER across all evaluation scenarios. The other models are trained on either clean speech or noisy speech, and they only achieve low error rates in the same noise conditions they were trained on.

When considering only models trained on clean speech (WSJ), the probabilistic models `P/WSJ/0` and `P/WSJ/1e-6` perform slightly better than the deterministic model `D/WSJ`, with an average CER reduction of 4.5% on test_eval92 and test_dev93. The model `P/WSJ/4e-6` is still competitive, but reports higher standard deviations due to higher noise level of the network.

Recent work on end-to-end models with deterministic weights and without external language models reports similar error rates. The models from recent work are represented by `Single-E2E` [58] (trained on tr05_simu_real, channel 5) and `ESPnet` [44] (trained on train_si284).

5.3 PRUNING EXPERIMENTS

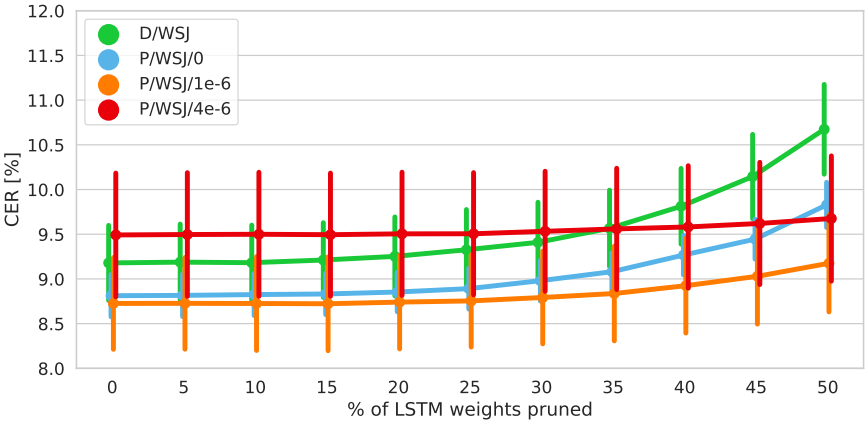
5.3.1 *Setup*

The pruning experiment is carried out on the models that were trained on WSJ train_si284, i.e. `D/WSJ`, `P/WSJ/0`, `P/WSJ/1e-6` and `P/WSJ/4e-6`. The LSTM weight parameters, which account for $> 99\%$ of the model parameters, are pruned while the rest of the model parameters is left unchanged. For the deterministic models, the LSTM weights w^{LSTM} are ordered by magnitude and the lowest X percent of magnitude weights are pruned, i.e. set to zero. For the probabilistic models, the LSTM mean weights μ^{LSTM} are ordered by SNR and the lowest X percent of SNR weight means are pruned. The models are tested on test_dev93 without any retraining after pruning. The probabilistic models use the LSTM mean weights μ^{LSTM} for testing.

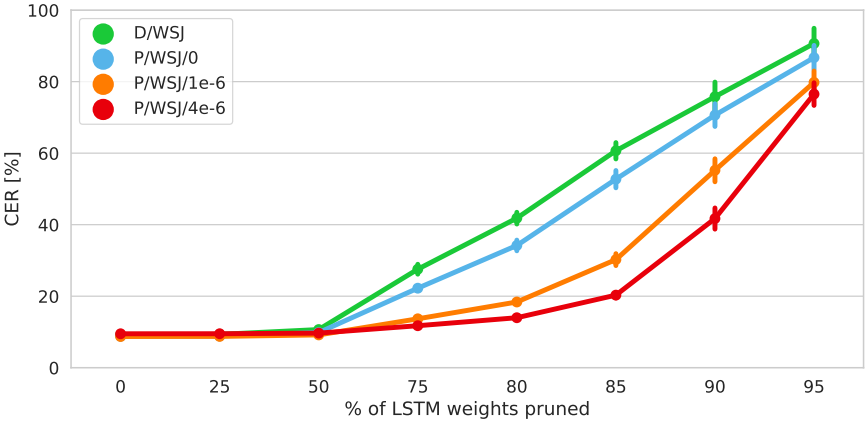
5.3.2 *Results*

The pruning results are reported in Figure 5.9. For sparsity levels smaller than 50% (Figure 5.9 (a)), all models achieve similar CER within the boundary of one standard deviation. For sparsity levels between 50% to 95% (Figure 5.9 (b)), all probabilistic models are able to achieve significantly lower error rates with the same sparsity level than the deterministic model `D/WSJ`.

Probabilistic models with lower SNR tolerate higher sparsity levels than models with higher SNR, and `P/WSJ/4e-6` tolerates the highest sparsity levels. With a 75% sparsity level, `P/WSJ/4e-6` achieves 11.7% CER, which is a relative CER reduction of 57.5% compared to the 27.5% CER of `D/WSJ`.



(a)



(b)

FIGURE 5.9: Weight pruning results when testing on the WSJ test_dev93 subset, for 0...50% sparsity (a) and 0...95% sparsity (b). The CER is given as average ± standard deviation over 25 runs. All probabilistic models show lower CER at similar sparsity levels than the deterministic model `D/WSJ`. The probabilistic model with the lowest SNR `P/WSJ/4e-6` shows the smallest CER increase under pruning.

5.4 DOMAIN ADAPTATION EXPERIMENTS

5.4.1 *Setup*

The adaptation experiment is carried out on the following models that were trained on train_si284: `D/WSJ`, `P/WSJ/0`, and `P/WSJ/4e-6`. The models were originally trained on clean speech data from WSJ, and now they are adapted to noisy speech data from the CHiME-4 dataset by further training on the dt05_real subset. The models are adapted for 25 epochs with the CTC loss $\mathcal{L}_{CTC}$ and the Adam optimizer (learning rate 1e-3). Note that the same number of epochs and the same learning rate is used for adaptation and baseline training. This strategy is different from conventional adaptation setups that use fine-tuning with fewer epochs and smaller learning rates for adaptation (e.g. [101]). In order to analyze the effect of Gaussian weights, we only adapt the weights w^{LSTM} (deterministic models) or the weight mean μ^{LSTM} (probabilistic models) of the LSTM cells. The biases b^{LSTM} and the projection weights w^{PROJ} are left unchanged.

As a measure to counter catastrophic forgetting, auxiliary loss terms are included that penalize the overwriting of weights. The penalties are different for deterministic and probabilistic models and their effect is plotted for a typical example in Figure 5.10. For the deterministic model, we propose an auxiliary L2 penalty $\mathcal{L}_{L2}$ between updated weight value w^{LSTM} and pre-adaptation weight value w^{LSTM*}:

$$\mathcal{L}_{L2} = (w^{LSTM} - w^{LSTM*})^2 \tag{5.12}$$

The L2 penalty prevents catastrophic forgetting by forcing weight updates to stay close to the original value. For the probabilistic models, we include an auxiliary SNR penalty $\mathcal{L}_{SNR}$ between updated mean weight value μ^{LSTM} and pre-adaptation mean weight value μ^{LSTM*}:

$$\mathcal{L}_{SNR} = \mathrm{SNR}(\mu^{LSTM} - \mu^{LSTM*})^2 \tag{5.13}$$

The inclusion of the SNR value in the loss term penalizes updates on parameters with higher SNR, which are assumed to be more important than lower SNR parameters to solve the original clean speech task. Note that besides the use of SNR information, the SNR penalty from Eq. (5.13) is similar to the L2 penalty from Eq. (5.12). The auxiliary penalties are scaled with the parameter λ_{aux} that is varied between $\{0, 0.1, ..., 727.9, 1000.0\}$ in 30 geometrically spaced steps, and the full loss function for adaptation is $\mathcal{L} = \mathcal{L}_{CTC} + \lambda_{aux}\mathcal{L}_{aux}$.

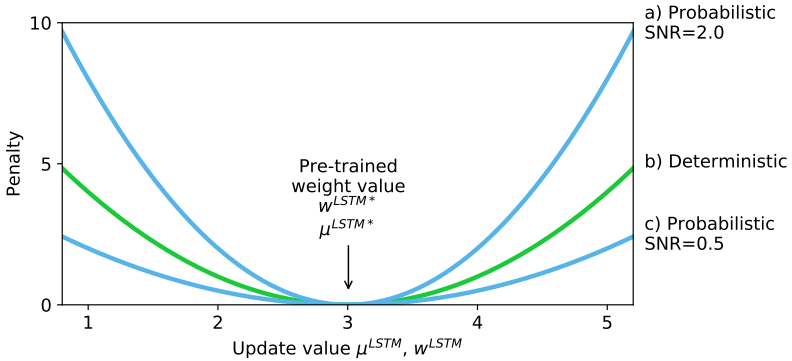


FIGURE 5.10: The plot shows the auxiliary L2 (green) and SNR (blue) penalties when overwriting a pre-trained weight value $w^{LSTM*}, \mu^{LSTM*} = 3$. The L2 penalty only depends on the difference between parameter and update value. The SNR penalty rescales the update cost with the SNR level. As a result, the higher SNR=2 parameter in curve a) is more costly to overwrite then the lower SNR=0.5 parameter in curve c). Therefore, the higher SNR parameter is more likely to be preservered during adaptation.

5.4.2 *Results*

The CER of the adapted models is evaluated for every epoch of adaptation on the subsets test_eval92 (clean speech) and et05_real (noisy speech), and the results are reported in Figure 5.11. The results show three tiers of adaptation characteristics. The trade-off between error rates on clean speech and noisy speech is smallest for the lower SNR model `P/WSJ/4e-6`, intermediate for the higher SNR model `P/WSJ/0` and highest for the deterministic model `D/WSJ`. In other words, `D/WSJ` is more prone to forgetting the original clean speech task than both probabilistic models, and a lower parameter SNR further reduces forgetting. When allowing for 8.0% CER on test_eval92, then `P/WSJ/4e-6` reaches 42.7% CER on et05_real, while `D/WSJ` reaches 50.2% CER, a relative reduction of 15%.

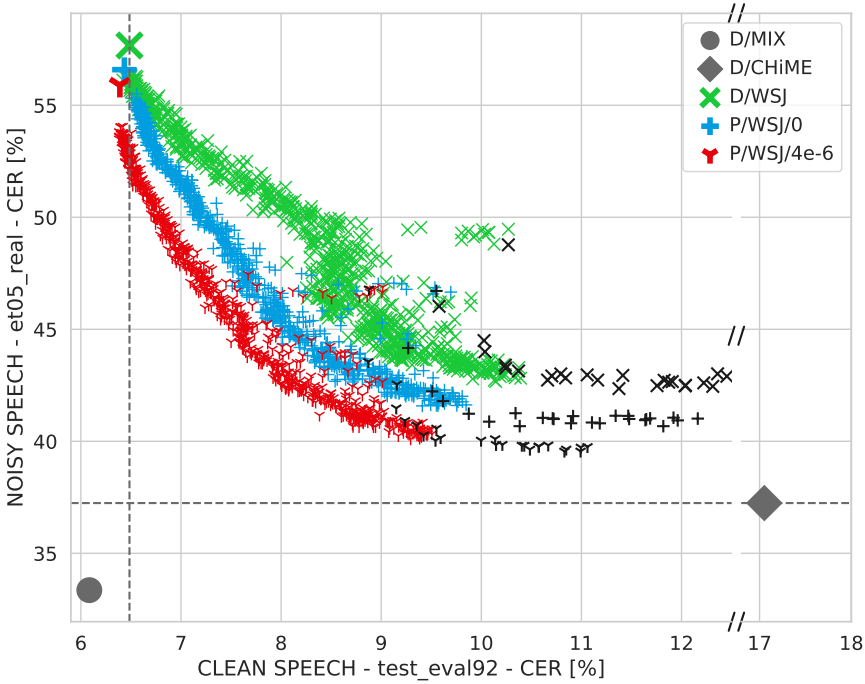


FIGURE 5.11: Adaptation results when adapting networks trained on clean speech, to noisy speech. Large dots show pre-adaptation error rates. Black dots denote models adapted with $\lambda_{aux} = 0$ (no additional penalty), while colored dots correspond to models with $\lambda_{aux} > 0$ (L2 or SNR penalty active). Every dot represents a different epoch of adaptation and a different λ_{aux}. The probabilistic networks show less forgetting on the original clean speech data during adaptation. The low SNR probabilistic model `P/WSJ/4e-6` shows the least amount of forgetting. © 2019 IEEE

5.5 CONCLUSION

This thesis chapter evaluated end-to-end models for ASR that use LSTM units with probabilistic weight parameters. The parameters are sampled from a Gaussian distribution with a parameter-specific mean and standard deviation. Despite the probabilistic formulation, the model is trainable with the same cost function as a model with deterministic parameters.

Experimental results show that probabilistic models achieved error rates on par or better than deterministic models. Despite an additional parameter sampling step, probabilistic parameters only lead to a 4% increase in training time compared to deterministic parameters. When pruning weights on an already trained model, the probabilistic models tolerated higher sparsity levels at lower error rates than the deterministic models. Also, during an adaptation experiment from clean to noisy speech, the probabilistic models showed less forgetting on the original clean speech task than deterministic models. A key advantage of probabilistic models is the availability of the parameter-specific SNR, which is highly correlated with the importance of a parameter for the task it was trained on. The parameter-specific SNR helped to identify less important parameters for pruning and to restrict updates on important parameters during adaptation.

The average SNR of the end-to-end model parameters is controllable by an additional loss term that enforces higher standard deviation. When comparing probabilistic models with different SNR levels, the results show that models with lower SNR exhibit improved pruning and adaptation characteristics. One reason for these findings might be that noisier parameters exhibit a higher degree of uncertainty, and therefore tolerate larger amounts of the parameter updates that appear in both pruning and adaptation. Future studies could evaluate probabilistic neural networks on larger speech datasets; and also with acoustic models that are based on neural network units other than LSTMs, e.g. CNNs.

6

CONCLUSION

6.1 SUMMARY

This thesis advances the state-of-the art in end-to-end models for ASR, with the overall goals of improving noise robustness and model interpretability. The thesis introduced three contributions.

First, a novel training method was proposed that improves the noise robustness of the end-to-end model with a curriculum learning strategy. The network training follows a SNR schedule that starts training at low SNR levels, and then gradually exposes the network to higher SNR levels. The curriculum training method did not require any changes in the neural network architecture. Compared to conventional multi-condition training methods, the curriculum learning strategy improved the recognition accuracy in noisy conditions.

The second contribution evaluated end-to-end models in the context of multi-modal, multi-sensor setups. To deal with multi-sensor input, the neural network architecture was modified to include a sensory attention mechanism. Despite adding relatively few parameters, the sensory attention mechanism allowed the network to extract information from multiple sensors and dynamically tune its attention towards less noisy sensors. In synthetic and real-world noisy conditions, the sensor tuning increased the accuracy and the model interpretability, as the attentional signal was highly correlated with the sensor noise level. Furthermore, the attention mechanism robustly dealt with adding and removing sensors after training. Compared to standard multi-sensor processing strategies such as concatenation and averaging, the attention mechanism performed on par or better across all experiments, with the additional benefit of identifying relative sensor SNR levels. Compared to conventional beamforming algorithms, the sensory attention mechanism resulted in higher error rates, but remained compatible with the end-to-end paradigm and significantly reduced the model complexity.

The third contribution introduced the most significant modification to the neural network architecture, replacing the deterministic network parameters with probabilistic parameters. The network parameters are sampled from probability distributions that are learned during training and exhibit a parameter-specific degree of uncertainty. The parameter uncertainty was

used as a proxy measure for parameter importance, greatly improving the interpretability of the millions of parameters found in end-to-end models. The importance information served as valuable information in a parameter pruning scenario for saved computation and a domain adaptation scenario for increased noise robustness.

6.2 APPLICATIONS AND OUTLOOK

The overall goals of this thesis were closely aligned with real-world applications of ASR systems. Modern ASR systems deal with noisy speech in various environments (e.g. in cars, conference rooms, living rooms etc.), and noise robustness is a key factor in avoiding customer frustrations. In these noisy conditions, many ASR systems rely on multi-sensor input and multi-sensor processing for improved accuracy. Prominent examples use multiple microphones (7 for Amazon Echo, 6 for Apple HomePod, 2 for Google Home), and multi-modal input configurations are actively explored (audio-visual conference transcription devices from Microsoft). Given the pervasiveness of multi-sensor input, the previous research focus of end-to-end models on single-channel input is limiting their usefulness in real-world applications. In this context, the proposed sensory attention mechanism brings end-to-end models closer to the application side. Beyond the plain accuracy improvement, the sensory attention mechanism also had two further positive implications that are relevant when dealing with physical sensor hardware. First, the attentional signal allowed to identify useful sensors with high SNR level. Second, the attention mechanism allowed to change the input configuration without any re-training, including sensor removal and addition. For real devices with sensor hardware, the attentional signal could help identifying failing sensors that need replacement or sub-optimal sensors that may be removed to save hardware and computation resources. From a more critical point of view, there still remains further potential to improve the sensory attention mechanism. The modality independent formulation might be useful for compatibility reasons, but also presents a limitation compared to more specialized algorithms. One such example is multi-channel audio input where beamforming algorithms use phase information for improved signal enhancement. In the current form, the sensory attention mechanism does not exploit phase information and future studies might improve on this aspect. Ultimately, this thesis contribution was only one of a few studies ([45, 59]) that evaluated end-to-end models for ASR in the context of multi-sensor input. Given the small amount of studies, it seems reasonable that there are

more datasets, network architectures and sensor combination strategies to explore.

This thesis also proposed a novel domain adaptation strategy for end-to-end models. Domain adaptation is an important tool to increase the accuracy of ASR systems on different speech domains, e.g. speech in different noisy conditions (cafe, airplane, office, living room etc.). End-to-end models are a particularly challenging scenario for domain adaptation, as the adaptation process overwrites network parameters. The overwriting process leads to catastrophic forgetting of the old domain. In practice, a network that was originally trained on clean speech has reduced accuracy on clean speech after adaptation to noisy speech. One could now introduce a separate set of domain-specific parameters to avoid catastrophic forgetting, but would then have to recognize the domain and select the corresponding parameter set during inference. Such an approach effectively creates a model ensemble. This thesis explored an alternative strategy for domain adaptation that is inspired by continual learning. From the view of continual learning, a single model is supposed to learn one task after the other, and to perform well on all tasks. In ASR, tasks can be considered as different speech domains. The single model is clearly superior compared to the ensemble, as the single model does not require to recognize the domain and then select the appropriate model. In the context of domain adaptation for end-to-end models in ASR, this contribution represents a first step with continual learning strategies. However, there remains significant potential for future studies and improvements on three levels. First, the proposed study only evaluated continual learning for two consecutive domains, but real-world applications might require to learn many more consecutive domains. Second, other continual learning scenarios on ASR, e.g. speaker adaptation, have not yet been explored in this study. Third, the domain adaptation strategy used the parameter SNR as a proxy measure for parameter importance. However, the parameter SNR only allowed to predict the effect of small, local changes on single parameters. In consequence, the network was forced to explore a constrained space around the parameter value during adaptation. Future work could explore strategies that allow to predict the effect of larger jumps in the parameter space and also the effect when groups of parameters are changed at the same time.

The previous two paragraphs have put the sensory attention mechanism and parameter uncertainty studies in the larger context of their respective research fields and applications. However, this thesis provides a unique occasion to compare the impact of two very different network modifications on the model interpretability. In fact, the model interpretability was enhanced on

two different levels: either at the *activation* level for sensory attention, or on the *parameter* level for parameter uncertainty. The activation level is particularly useful for *model verification*, i.e. checking if a model is doing something that is intuitively reasonable. The key to activation level interpretability is to build a network architecture that produces interpretable activations at intermediate computation steps. The sensory attention mechanism follows this design rule by computing activations that are interpretable as sensory attention weights. However, the activations are no defining property of a neural network, and they can only represent intermediate computation results that are generated by the interaction of a data sample and a neural network. This means that activation-level methods cannot increase the interpretability of neural network parameters alone. To gain more insights on the parameter level, this thesis proposed to use parameter uncertainty as a proxy measure for parameter importance. With parameter uncertainty, the parameters are now interpretable without any further computation on data samples, and by looking at isolated parameters alone. The parameter-specific uncertainty allowed to predict the effect of parameter changes, and therefore to modify a neural network in a selective fashion during pruning and adapation. Based on this argumentation, the parameter level is particularly useful for *model modification*. Given the different use cases of model verification and modification, it remains the decision of the end user to select the appropriate network modification. In theory, both approaches should also be compatible, and future work could attempt to use both approaches for increased model interpretability at the activation and parameter level.

Finally, all gains in model interpretability that were achieved in this thesis required a modification in the neural network. While the activation level gains could be achieved with standard neural network units and default deterministic parameters, the parameter level gains required to entirely change the neural network definition. Deterministic parameters are inherently limited in interpretability, as they only provide isolated values without any further information. The parameter uncertainty study used probabilistic parameters and could at least give some information on the distribution of parameters. This implies that future improvements in parameter interpretability may have to deprecate deterministic parameters, and further explore probabilistic parameters in the context of Bayesian neural networks [66] or generative models for parameter sampling [102].

A

APPENDIX

A.1 VALID ALIGNMENTS IN CTC

The CTC loss considers all valid alignments from the length-T^* network output $\mathbf{h}$ to the length-U target sequence $\mathbf{y}$. The number of valid alignments increases quickly with T^*: for the word `CAT`, i.e. $\mathbf{y} = \{C, A, T\}$ and $U = 3$, the number of alignments is 28 for $T^* = 5$; 1716 for $T^* = 10$; and 1.43x10e-9 for $T^* = 100$. This section discusses the combinatorics to compute the number of valid alignments for the simplified case when there are no repetitions in the target sequence.

We assume a length-U target sequence $\mathbf{y} = \{y_1, ..., y_U\}$ of text tokens y_u without repeated elements [1], and a length-T^* alignment $\mathbf{a} = \{a_1, ..., a_{T^*}\}$ of text tokens $a_\tau \in \{y_1, ..., y_U, \epsilon\}$, with the blank token ϵ. An alignment $\mathbf{a}$ is considered valid when it can be mapped to the target sequence $\mathbf{y}$ by the mapping function $\mathcal{A} : \mathbf{a} \mapsto \mathbf{y}$. The number of valid alignments, N_{val}, is defined by the binomial coefficient following Eq. (A.1). While this formula has been presented before in [103], no derivation was given. For the more complex case of repeated labels, [69] gives a formula, but also without derivation. The following paragraphs describe the derivation of Eq. (A.1) in 4 steps.

$$N_{val} = \binom{T^* + U}{T^* - U} \tag{A.1}$$

STEP 1. We first consider all valid mappings from sequences without blank labels. For $y = \{C, A, T\}$ and $T^* = 5$, two example alignments would be $\{C, A, A, A, T\}$ or $\{C, C, A, A, T\}$. In other words, every time step $\tau \in \{1, ..., T^*\}$ is assigned a non-blank token $y_u \in \mathbf{y}$. Because only valid mappings are considered, every token y_u has to occur at least once, e.g. the alignment $\{A, A, A, T, T\}$ would be non-valid. This corresponds to finding a solution to the equation $y_1 + ... + y_U = T^*$, where every element $y_u \geq 1$. This is solved with a *stars-and-bars* approach [104], and the number of alignments in step 1 is defined in Eq. (A.2):

1 For example, with character tokens this includes targets such as `CAT` or `BEHIND`, but not `HELLO` or `PENDING`.

$$N^I = \binom{T^* - U + U - 1}{U - 1} = \binom{T^* - 1}{U - 1}. \tag{A.2}$$

STEP 2. We now consider alignments with blank tokens. The number of blank tokens $b = T^* - U - k$ is the number of time steps T^* minus the number of individual target tokens U and minus the number of multiple token occurrences k. In this intermediate step, only combinations are considered, i.e. permutations such $\{C, A, A, T, \epsilon\}$ and $\{\epsilon, C, A, A, T\}$ are counted only once. We sum over all alignments from $k = 0$ (e.g. $\{C, A, T, \epsilon, \epsilon\}$) to $k = T^* - U$ (e.g. $\{C, A, A, A, T\}$) as defined in Eq. (A.3):

$$N^{II} = \sum_{k=0}^{T^*-U} \binom{k + U - 1}{U - 1} = \binom{T^*}{U} \tag{A.3}$$

STEP 3. The intermediate step 2 did not differentiate between paths with different positions of blank token insertions, i.e. permutations were only counted once. In this step, we consider these permutations. Recall that every alignment has the length T^*. For every sequence of length T^*, we have to pick the positions of $b = T^* - U - k$ blank tokens. This corresponds to the b-permutations without repetition in a set of T^* elements, and the number of valid alignments is now defined in Eq. (A.4)

$$N^{III} = \sum_{k=0}^{T^*-U} \underbrace{\binom{k + U - 1}{U - 1}}_{\text{combinations}} \underbrace{\binom{T^*}{T^* - U - k}}_{\text{permutations}} \tag{A.4}$$

However, Eq. (A.4) allows blank insertion that are non-valid, i.e. $\{C, A, \epsilon, A, T\}$, and can therefore only be used as an estimate for the upper bound of valid alignments.

STEP 4. We correct Eq. (A.4) to only include the count of valid blank permutations. Assume the exemplary non-blank combination $\{C, C, A, A, T\}$. Possible blank insertion buckets are denoted with an underscore $'\_'$, resulting in $\{\_, C, \_, C, \_, A, \_, A, \_, T, \_\}$. The number of buckets corresponds to $B = U + k + 1$. Note that each bucket can be filled with multiple blank tokens if $b = T^* - U - k$ is sufficiently high. We find that out of B buckets, only $B_{val} = U + 1$ buckets accept blank tokens without making the alignment non-valid. In the example above, denoting valid buckets with $'+'$ and non-

valid buckets with $'-'$ would generate $\{+, C, -, C, +, A, -, A, +, T, +\}$. The stars-and-bars method yields Eq. (A.5) for valid blank permutations:

$$\binom{b + B_{val}}{U} = \binom{T^* - U - k + U}{U} = \binom{T^* - k}{U}. \tag{A.5}$$

Replacing the permutations term in Eq. (A.5) yields the accurate number of valid alignments defined in Eq. (A.6):

$$N_{val} = \sum_{k=0}^{T^*-U} \binom{k + U - 1}{U - 1} \binom{T^* - k}{U} \tag{A.6}$$

The relation in Eq. (A.6) may be simplified by identity 5.26 from [105] to match Eq. (A.1).

A.2 MULTI-CHANNEL ATTENTION EXAMPLES

In the following section, the `STAN-2CH` attention weights and merged representations for 3 additional samples from the subsets dt05_real and et05_real of the CHiME-4 dataset are plotted. The samples are summarized in Table A.1, with corrupted channels given after visual and listening inspection.

Sample key	Subset	Corrupted channels	Figure
M03_22HC010P_STR	dt05_real	2	Figure A.1
M03_423C020N_PED	dt05_real	None	Figure A.2
F05_441C020G_STR	et05_real	2, 4	Figure A.3

TABLE A.1: Sample keys and corrupted channels (based on visual inspection and listening tests), ordered by the number of corrupted channels.

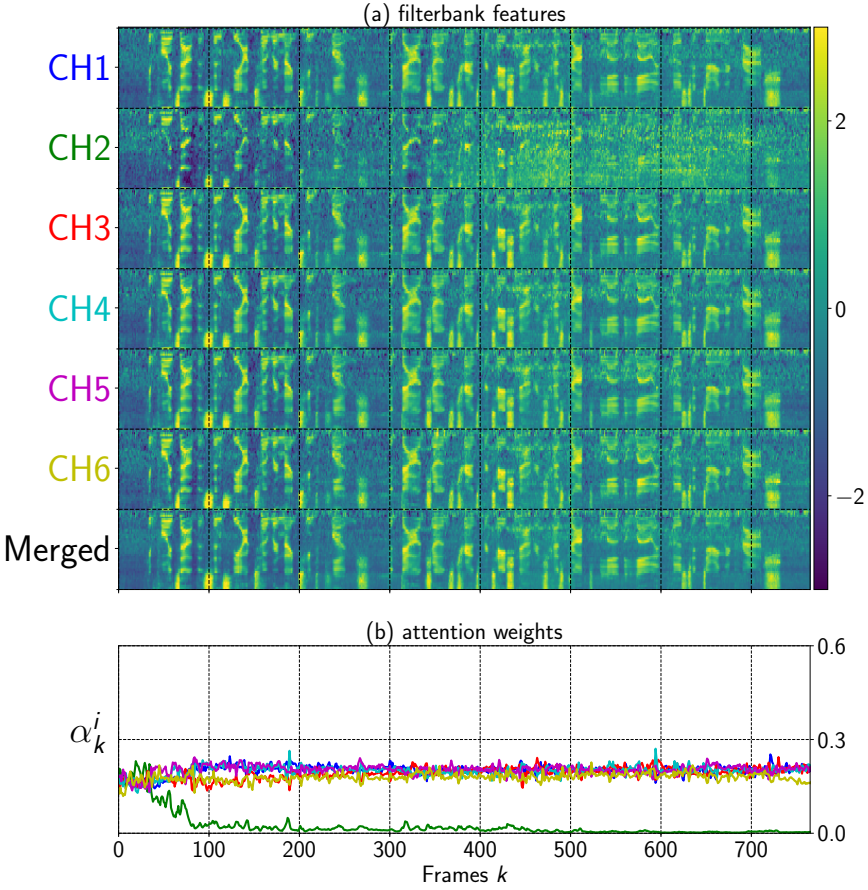


FIGURE A.1: Corrupted channels - ch2. This sample is representative for most of the real noisy data: the backward channel 2 is slightly corrupted, while the other channels seem similar. The STAN model assigns lower attention weights to channel 2.

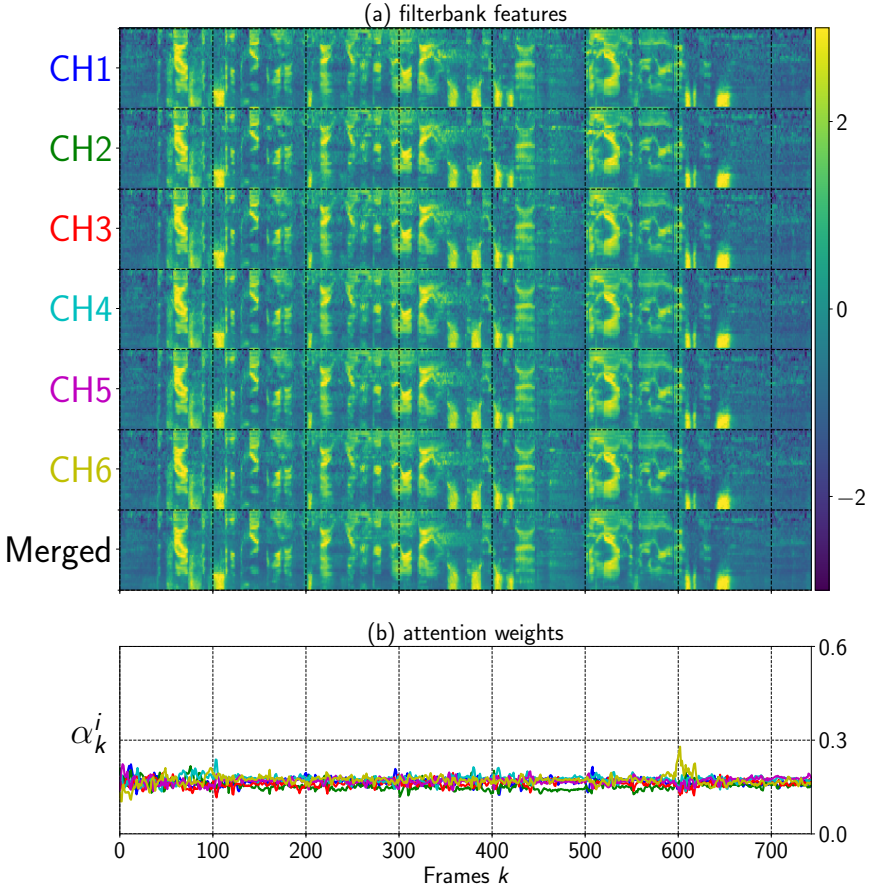


FIGURE A.2: Corrupted channels - none. This sample shows the attention response when no channel is corrupted. This represents a rare case in the dataset where the backwards channel 2 delivers similar quality to the front channels. In consequence, the STAN model computes similar attention weights for all channels.

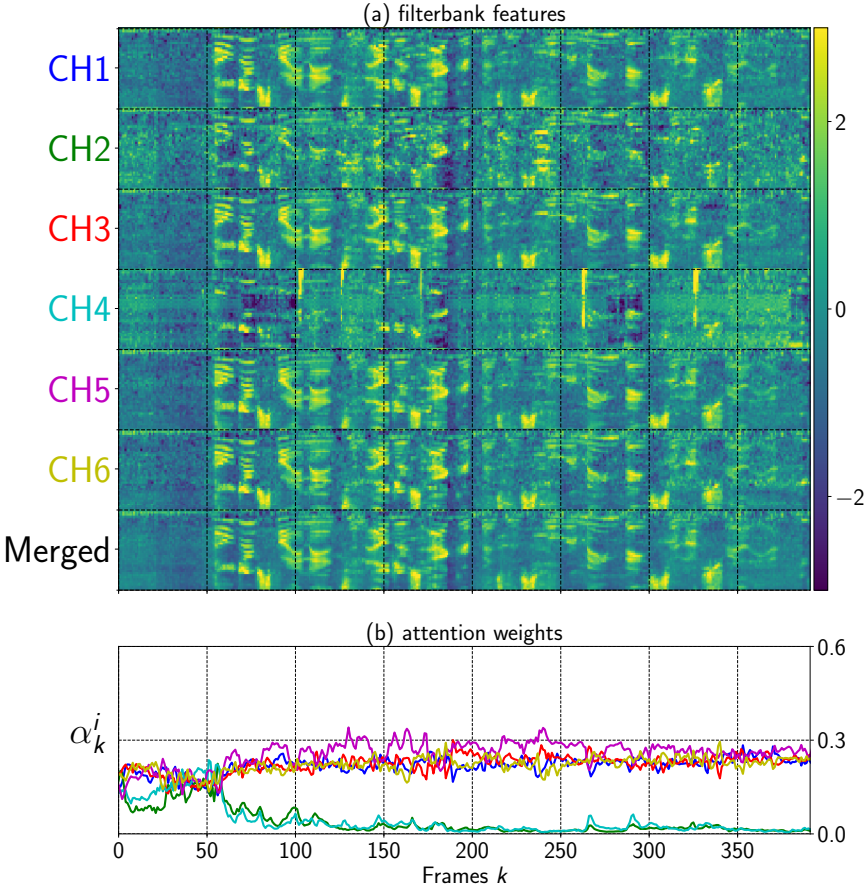


FIGURE A.3: Corrupted channels - 2, 4. Channel 2 is the backwards channel with typically lower SNR and reduced attention weights. After listening inspection, channel 4 sounds like there was a technical recording issue for this sample. On both noisy channels, STAN reduces the attention weights.

BIBLIOGRAPHY

[1] T. Poibeau, *Machine Translation*. MIT Press, 2017.

[2] P. Taylor, *Text-to-speech Synthesis*. Cambridge University Press, 2009.

[3] L. Rabiner and B.-H. Juang, "Historical Perspective of the Field of ASR/NLU", in *Springer Handbook of Speech Processing*, Springer, 2008, pp. 521–538.

[4] R. Pieraccini, *The Voice in the Machine: Building Computers that Understand Speech*. MIT Press, 2012.

[5] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic Recognition of Spoken Digits", *The Journal of the Acoustical Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[6] T. K. Vintsyuk, "Speech Discrimination by Dynamic Programming", *Cybernetics and Systems Analysis*, vol. 4, no. 1, pp. 52–57, 1968.

[7] V. M. Velichko and N. G. Zagoruyko, "Automatic Recognition of 200 Words", *International Journal of Man-Machine Studies*, vol. 2, no. 3, pp. 223–234, 1970.

[8] B. T. Lowerre, "The Harpy Speech Recognition System.", Pittsburgh, PA, USA, Tech. Rep., 1976.

[9] J. Baker, "The DRAGON System–An Overview", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 24–29, 1975.

[10] R. Bakis, "Continuous Speech Recognition via Centisecond Acoustic States", *The Journal of the Acoustical Society of America*, vol. 59, no. S1, p. 97, 1976.

[11] F. Jelinek, "Continuous Speech Recognition by Statistical Methods", *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532–556, 1976.

[12] L. R. Bahl, R. Bakis, J. Bellegarda, P. F. Brown, D. Burshtein, S. K. Das, P. V. de Souza, P. S. Gopalakrishnan, F. Jelinek, D. Kanevsky, R. L. Mercer, A. J. Nadas, D. Nahamoo, and M. A. Picheny, "Large vocabulary natural language continuous speech recognition", in *1989 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1989, 465–467 vol.1.

[13] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1994.

[14] R. P. Lippmann, "Review of Neural Networks for Speech Recognition", *Neural Computation*, vol. 1, no. 1, pp. 1–38, 1989.

[15] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups", *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[16] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "RWTH ASR Systems for LibriSpeech: Hybrid vs. Attention", *Interspeech 2019*, to appear, 2019.

[17] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[18] A. Graves and N. Jaitly, "Towards End-to-end Speech Recognition with Recurrent Neural Networks", in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, 2014, pp. 1764–1772.

[19] A. Hannun *et al.*, "Deep Speech: Scaling up End-to-end Speech Recognition", *arXiv preprint arXiv:1412.5567*, 2014.

[20] D. Amodei *et al.*, "Deep Speech 2 : End-to-end Speech Recognition in English and Mandarin", in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, vol. 48, 2016, pp. 173–182.

[21] E. Battenberg, J. Chen, R. Child, A. Coates, Y. G. Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring Neural Transducers for End-to-end Speech Recognition", in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 206–213.

[22] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end Speech Recognition using Deep RNN Models and WFST-based Decoding", in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174.

[23] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition", in *Advances in Neural Information Processing Systems 28*, 2015, pp. 577–585.

[24] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end Attention-based Large Vocabulary Speech Recognition", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949.

[25] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A Neural Network for Large Vocabulary Conversational Speech Recognition", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[26] L. Lu, X. Zhang, and S. Renais, "On Training the Recurrent Neural Network Encoder-decoder for Large Vocabulary End-to-end Speech Recognition", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5060–5064.

[27] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based End-to-end Speech Recognition using Multi-task Learning", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4835–4839.

[28] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks", in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.

[29] A. Graves, "Sequence Transduction with Recurrent Neural Networks", *arXiv preprint arXiv:1211.3711*, 2012.

[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", *arXiv preprint arXiv:1409.0473*, 2014.

[31] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, *Computer, Speech & Language*,

[32] S. Watanabe, M. Delcroix, F. Metze, and J. R. Hershey, *New Era for Robust Speech Recognition: Exploiting Deep Learning*. Springer International Publishing, 2017.

[33] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. Springer Science & Business Media, 2008, vol. 1.

[34] M. Brandstein and D. Ward, *Microphone Arrays: Signal Processing Techniques and Applications*. Springer Science & Business Media, 2013.

[35] S. Boll, "Suppression of Acoustic Noise in Speech Using Spectral Subtraction", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[36] Y. Fujita, R. Takashima, T. Homma, R. Ikeshita, Y. Kawaguchi, T. Sumiyoshi, T. Endo, and M. Togami, "Unified ASR System Using LGM-based Source Separation, Noise-robust Feature Extraction, and Word Hypothesis Selection", in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 416–422.

[37] T. Hori, Z. Chen, H. Erdogan, J. R. Hershey, J. Le Roux, V. Mitra, and S. Watanabe, "The MERL/SRI System for the 3RD CHiME Challenge Using Beamforming, Robust Feature Extraction, and Advanced Speech Recognition", in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 475–481.

[38] M. J. F. Gales, "Maximum Likelihood Linear Transformations for HMM-based Speech Recognition", *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[39] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "Compensation for the Effect of the Communication Channel in Auditory-like Analysis of Speech (RASTA-PLP)", in *2nd European Conference on Speech Communication and Technology*, 1991.

[40] D. Palaz, M. Magimai-Doss, R. Collobert, *et al.*, "Analysis of CNN-based Speech Recognition System Using Raw Speech as Input", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2015.

[41] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu Features? End-to-end Speech Emotion Recognition Using a Deep Convolutional Recurrent Network", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5200–5204.

[42] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end Speech Recognition from the Raw Waveform", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 781–785.

[43] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning Filterbanks from Raw Speech for Phone Recognition", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5509–5513.

[44] 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 *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 2207–2211.

[45] T. Ochiai, S. Watanabe, T. Hori, J. R. Hershey, and X. Xiao, "Unified Architecture for Multichannel End-to-end Speech Recognition With Neural Beamforming", *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1274–1288, 2017.

[46] M. L. Seltzer, D. Yu, and Y. Wang, "An Investigation of Deep Neural Networks for Noise Robust Speech Recognition", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7398–7402.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[48] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009, pp. 41–48.

[49] S. Renals and P. Swietojanski, "Neural Networks for Distant Speech Recognition", in *2014 4th Joint Workshop on Hands-free Speech Communication and Microphone Arrays (HSCMA)*, 2014, pp. 172–176.

[50] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional Neural Networks for Distant Speech Recognition", *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1120–1124, 2014.

[51] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, M. Bacchiani, and Andrew, "Speaker Location and Microphone Spacing Invariant Acoustic Modeling from Raw Multichannel Waveforms", in *2015 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2015, pp. 30–36.

[52] Y. Liu, P. Zhang, and T. Hain, "Using Neural Network Front-ends on Far Field Multiple Microphones Based Speech Recognition", in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5542–5546.

[53] X. Xiao, S. Watanabe, H. Erdogan, L. Lu, J. Hershey, M. L. Seltzer, G. Chen, Y. Zhang, M. Mandel, and D. Yu, "Deep Beamforming Networks for Multi-channel Speech Recognition", in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5745–5749.

[54] H. Erdogan *et al.*, "Multi-channel Speech Recognition: LSTMs All the Way Through", in *CHiME-4 workshop*, 2016.

[55] B. Li, T. N. Sainath, R. J. Weiss, K. W. Wilson, and M. Bacchiani, "Neural Network Adaptive Beamforming for Robust Multichannel Speech Recognition", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2016, pp. 1976–1980.

[56] Z. Meng, S. Watanabe, J. R. Hershey, and H. Erdogan, "Deep Long Short-term Memory Adaptive Beamforming Networks for Multichannel Robust Speech Recognition", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 271–275.

[57] J. Heymann, L. Drude, C. Boeddeker, P. Hanebrink, and R. Haeb-Umbach, "Beamnet: End-to-end Training of a Beamformer-supported Multi-channel ASR System", in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5325–5329.

[58] S. Kim and I. Lane, "Recurrent Models for Auditory Attention in Multi-Microphone Distant Speech Recognition", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2016, pp. 3838–3842.

[59] S. Kim and I. Lane, "End-to-end Speech Recognition with Auditory Attention for Multi-Microphone Distance Speech Recognition", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017, pp. 3867–3871.

[60] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip Reading Sentences in the Wild", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3444–3453.

[61] S. Petridis, T. Stafylakis, P. Ma, F. Cai, G. Tzimiropoulos, and M. Pantic, "End-to-end Audiovisual Speech Recognition", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6548–6552.

[62] T. Afouras, J. S. Chung, and A. Zisserman, "The Conversation: Deep Audio-Visual Speech Enhancement", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 3244–3248.

[63] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to Listen at the Cocktail Party: A Speaker-independent Audio-visual Model for Speech Separation", *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, 112:1–112:11, 2018.

[64] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional Two-Stream Network Fusion for Video Action Recognition", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1933–1941.

[65] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem", in *Psychology of Learning and Motivation*, vol. 24, Elsevier, 1989, pp. 109–165.

[66] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 1613–1622.

[67] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian Recurrent Neural Networks", *arXiv preprint arXiv:1704.02798*, 2017.

[68] A. Graves, "Practical Variational Inference for Neural Networks", in *Advances in Neural Information Processing Systems 24*, 2011, pp. 2348–2356.

[69] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks", PhD thesis, Technical University Munich, 2008.

[70] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals", in *Soviet physics doklady*, vol. 10, 1966, pp. 707–710.

[71] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein, "Streaming End-to-end Speech Recognition for Mobile Devices", in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6381–6385.

[72] S. Braun, D. Neil, and S.-C. Liu, "A Curriculum Learning Method for Improved Noise Robustness in Automatic Speech Recognition", in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 548–552, DOI: 10.23919/EUSIPCO.2017.8081267.

[73] S. Yin, C. Liu, Z. Zhang, Y. Lin, D. Wang, J. Tejedor, T. F. Zheng, and Y. Li, "Noisy Training for Deep Neural Networks in Speech Recognition", *EURASIP Journal on Audio, Speech, and Music Processing*, no. 1, pp. 1–14, 2015.

[74] P. Koistinen and L. Holmström, "Kernel Regression and Backpropagation Training With Noise", in *Advances in Neural Information Processing Systems 4*, 1992, pp. 1033–1039.

[75] K. J. Geras and C. A. Sutton, "Scheduled Denoising Autoencoders", in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[76] J. Garofalo, D. Graff, D. Paul, and D. Pallett, "CSR-I (WSJ0) Complete, LDC93S6A", *Linguistic Data Consortium, Philadelphia*, 2007.

[77] Audacity developers. (2016). Audacity, [Online]. Available: `http://www.audacityteam.org/`.

[78] A. Varga and H. J. M. Steeneken, "Assessment for Automatic Speech Recognition: II. NOISEX-92: A Database and an Experiment to Study the effect of Additive Noise on Speech recognition systems", *Speech Communication*, vol. 12, no. 3, pp. 247–251, 1993.

[79] Lasagne developers. (2015). Lasagne: First release., [Online]. Available: `https://github.com/Lasagne/Lasagne/releases/tag/v0.1`.

[80] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[81] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks", in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, 2010, pp. 249–256.

[82] D. P. Kingma and J. Ba, "Adam: {A} Method for Stochastic Optimization", in *3rd International Conference on Learning Representations(ICLR)*, 2015.

[83] S. Braun, D. Neil, J. Anumula, E. Ceolini, and S.-C. Liu, "Attention-driven Multi-sensor Selection", in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8, DOI: 10.1109/IJCNN.2019.8852396.

[84] S. Braun, D. Neil, J. Anumula, E. Ceolini, and S.-C. Liu, "Multi-channel Attention for End-to-end Speech Recognition", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 17–21, DOI: 10.21437/Interspeech.2018-1301.

[85] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks", in *Advances in Neural Information Processing Systems 30*, 2017, pp. 971–980.

[86] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[87] M. Cooke, J. Barker, S. Cunningham, and X. Shao, "An Audio-visual Corpus for Speech Perception and Automatic Speech Recognition", *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2421–2424, 2006.

[88] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "LipNet: Sentence-level Lipreading", *arXiv preprint arXiv:1611.01599*, 2016.

[89] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 Faces In-The-Wild Challenge: Database and Results", *Image and Vision Computing*, vol. 47, pp. 3–18, 2016.

[90] X. Anguera, C. Wooters, and J. Hernando, "Acoustic Beamforming for Speaker Diarization of Meetings", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2011–2022, 2007.

[91] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization", *arXiv preprint arXiv:1607.08022*, 2016.

[92] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks", in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323.

[93] S. Braun and S.-C. Liu, "Parameter Uncertainty for End-to-end Speech Recognition", in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5636–5640, DOI: 10.1109/ICASSP.2019.8683066.

[94] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating Second-Order Functional Knowledge for Better Option Pricing", in *Advances in Neural Information Processing Systems 13*, 2001, pp. 472–478.

[95] D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick", in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2575–2583.

[96] G. E. Hinton and D. van Camp, "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights", in *Proceedings of the 6th Annual Conference on Computational Learning Theory (COLT)*, ACM, 1993, pp. 5–13.

[97] K.-C. Jim, C. L. Giles, B. G. Horne, *et al.*, "An Analysis of Noise in Recurrent Neural Networks: Convergence and Generalization", *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1424–1438, 1996.

[98] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light Gated Recurrent Units for Speech Recognition", *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, 2018.

[99] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic Differentiation in PyTorch", in *NIPS Autodiff Workshop*, 2017.

[100] S. Braun, "LSTM Benchmarks for Deep Learning Frameworks", *arXiv preprint arXiv:1806.01818*, 2018.

[101] T. Ochiai, S. Watanabe, S. Katagiri, T. Hori, and J. Hershey, "Speaker Adaptation for Multichannel End-to-end Speech Recognition", in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6707–6711.

[102] D. Ha, A. M. Dai, and Q. V. Le, "HyperNetworks", in *5th International Conference on Learning Representations (ICLR)*, 2017.

[103] A. Hannun, "Sequence Modeling with CTC", *Distill*, 2017.

[104] K. H. Rosen and K. Krithivasan, *Discrete Mathematics and its Applications: With Combinatorics and Graph Theory*. Tata McGraw-Hill Education, 2012.

[105] R. L. Graham, D. E. Knuth, O. Patashnik, and S. Liu, "Concrete Mathematics: A Foundation for Computer Science", *Computers in Physics*, vol. 3, no. 5, pp. 106–107, 1989.

CURRICULUM VITAE

PERSONAL DATA

Name	Stefan Braun
Date of Birth	March 10, 1989
Place of Birth	Oberkirch, Germany
Citizen of	Germany

EDUCATION

2015 – 2019	PhD candidate at the Institute of Neuroinformatics, Department of Information Technology and Electrical Engineering, *ETH Zürich, Zürich, Switzerland*
2013 – 2015	M. Sc. in Mechanical Engineering, *Karlsruher Institut für Technologie, Karlsruhe, Germany*
2011 – 2015	Diploma in Mechanical Engineering, *Institut National des Sciences Appliquées de Lyon, Lyon, France*
2009 – 2013	B. Sc. in Mechanical Engineering, *Karlsruher Institut für Technologie, Karlsruhe, Germany*
1999 – 2008	Abitur *Oken Gymnasium, Offenburg, Germany*

EMPLOYMENT

2015 – 2019	Scientific Researcher at the Institute of Neuroinformatics, *University of Zürich, Zürich, Switzerland*

PUBLICATIONS

Peer-reviewed contributions as first author:

[1] S. Braun, D. Neil, and S.-C. Liu, "A Curriculum Learning Method for Improved Noise Robustness in Automatic Speech Recognition", in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 548–552.

[2] S. Braun, D. Neil, J. Anumula, E. Ceolini, and S.-C. Liu, "Multi-channel Attention for End-to-end Speech Recognition", in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 17–21.

[3] S. Braun, D. Neil, J. Anumula, E. Ceolini, and S.-C. Liu, "Attention-driven Multi-sensor Selection", in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[4] S. Braun and S.-C. Liu, "Parameter Uncertainty for End-to-end Speech Recognition", in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5636–5640.

Peer-reviewed contributions as co-author:

[1] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S.-C. Liu, "Real-Time Speech Recognition for IoT Purpose using a Delta Recurrent Neural Network Accelerator", in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019.

[2] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S.-C. Liu, "Live Demonstration: Real-Time Spoken Digit Recognition using the DeltaRNN Accelerator", in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019.

[3] E. Ceolini, J. Anumula, S. Braun, and S.-C. Liu, "Event-driven Pipeline for Low-latency Low-compute Keyword Spotting and Speaker Verification System", in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7953–7957.