Performance Monitoring for End-to-End Speech Recognition

Ruizhi Li[1], *Gregory Sell*[1,2], *Hynek Hermansky*[1,2]

[1]Center for Language and Speech Processing, The Johns Hopkins University, USA
[2]Human Language Technology Center of Excellence, The Johns Hopkins University, USA
{ruizhili, gsell2, hynek}@jhu.edu

Abstract

Measuring performance of an automatic speech recognition (ASR) system without ground-truth could be beneficial in many scenarios, especially with data from unseen domains, where performance can be highly inconsistent. In conventional ASR systems, several performance monitoring (PM) techniques have been well-developed to monitor performance by looking at tri-phone posteriors or pre-softmax activations from neural network acoustic modeling. However, strategies for monitoring more recently developed end-to-end ASR systems have not yet been explored, and so that is the focus of this paper. We adapt previous PM measures (Entropy, M-measure and Auto-encoder) and apply our proposed RNN predictor in the end-to-end setting. These measures utilize the decoder output layer and attention probability vectors, and their predictive power is measured with simple linear models. Our findings suggest that decoder-level features are more feasible and informative than attention-level probabilities for PM measures, and that M-measure on the decoder posteriors achieves the best overall predictive performance with an average prediction error 8.8%. Entropy measures and RNN-based prediction also show competitive predictability, especially for unseen conditions.

Index Terms: End-to-End Speech Recognition, Performance Monitoring

1. Introduction

In recent years, significant improvement for conventional automatic speech recognition (ASR) has been achieved via advancements with Deep Neural Networks (DNNs). The main paradigm for an ASR system is the so-called hybrid approach, which involves training a DNN to predict context dependent phoneme states (or senones) from the acoustic features. However, if test data comes from a very different domain than DNN training data, it is possible for the recognizer to fail without any warning, e.g., confident prediction of incorrect labels. Predicting these failures is the goal of the work that follows. Humans are often aware of the uncertainty of decisions they are making [1]. Performance monitoring (PM) techniques aim for the same goal - to determine the quality of a system's output - based only on the behavior of the system and without any knowledge of the underlying truth. An effective PM measure could be useful in a number of applications [2, 3, 4, 5, 6, 7, 8, 9], such as multi-stream selection scenario [3, 4, 5, 6, 7, 8, 9] or semi-supervised training [2].

Unlike conventional ASR, end-to-end speech recognition approaches are designed to directly output word or character sequences from the input audio signal. This model subsumes several disjoint components in the hybrid ASR model (acoustic model, pronunciation model, language model) into a single neural network. As a result, all the components of an end-to-end model can be trained jointly to optimize a single objective. Two dominant end-to-end architectures for ASR are Connectionist Temporal Classification (CTC) [10, 11, 12] and attention-based encoder-decoder models [13, 14] . A joint CTC/Attention framework was proposed in [15, 16, 17] to take advantage of both architectures within a multi-task scheme. The joint model was shown to provide state-of-the-art end-to-end results for several benchmark datasets [17, 18].

This paper aims to explore the PM techniques applicable for an end-to-end framework. In the hybrid approach, tri-phone posterior distributions and their corresponding pre-softmax activations are typically treated as PM features. Averaged entropy over temporal frames was proposed as a confidence measure in stream-selection [3, 19]. Mean temporal distance on posteriors estimates the performance by capturing the divergence of any two frames over several time spans [20, 21]. Reconstruction error of an auto-encoder trained on pre-softmax features was also used as the selection criterion in a multi-stream system [22, 23].

In the end-to-end setting, there are two levels of probability distributions: attention weights and decoder posteriors. Instead of temporal posteriors in the conventional case, each probability distribution corresponds to a character-level prediction. Therefore, we must adapt the techniques used for hybrid systems to the joint CTC/attention model. Moreover, inspired by the success of discriminatively-trained DNNs, we propose using a Recurrent Neural Network (RNN) regression model trained to directly predict performance. Our analyses demonstrate strong correlations between PM measures and true performance, indicating that end-to-end ASR systems are indeed amenable to effective monitoring.

This paper is organized as follows: Section 2 explains the joint CTC/Attention model. The description of data configuration is in Section 3. Experiments with results and analyses are presented in Section 4. Finally, we conclude in Section 5.

2. End-to-End Architecture

The joint CTC/Attention model is designed to directly map T-length acoustic features $X = \{\mathbf{x}_t \in \mathbb{R}^D | t = 1, 2, ..., T\}$ in D dimensional space to an L-length letter sequence $C = \{c_l \in \mathcal{U} | l = 1, 2, ..., L\}$ where $\mathcal{U}$ is a set of distinct letters. The attention-based structure solves the ASR problem as a sequence mapping by using an encoder-decoder architecture. Joint training with CTC is added to help enforce temporally monotonic behavior in the attention alignments.

The overall end-to-end architecture is shown in Fig. 1. The encoder is shared by both the attention and CTC networks. Bidirectional Long Short-Term Memory (BLSTM) layers are utilized to model the temporal dependencies of the input sequence. The frame-wise hidden vector $\mathbf{h}_t$ at frame t is derived by encoding the full input sequence X:

$$\mathbf{h}_t = \text{Encoder}(X) \tag{1}$$

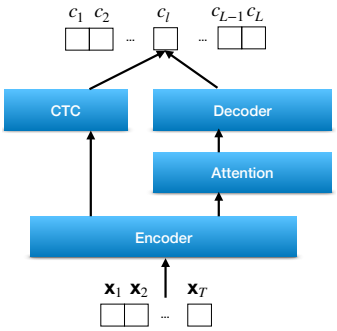


Figure 1: *Joint CTC/Attention End-to-End Architecture*

In the attention-based network, the letter-wise context vector $\mathbf{r}_l$ is formed as a weighted summation of frame-wise hidden vectors $\mathbf{h}_t$ using a content-based attention mechanism:

$$\mathbf{r}_l = \sum_{t=1}^{T} a_{lt}\mathbf{h}_t,\ a_{lt} = \text{ContentAttention}(\mathbf{q}_{l-1}, \mathbf{h}_t) \quad (2)$$

where $\mathbf{q}_{l-1}$ is the previous decoder state, and a_{lt} is the attention weight, which can be considered a soft-alignment of $\mathbf{h}_t$ for c_l. An LSTM-based decoder outputs the character-level conditional probability distribution $p(c_l|c_1, ..., c_{l-1}, X)$ for multi-task training and joint decoding.

During inference, the joint CTC/Attention model performs a label-synchronous beam search, which jointly predicts the next character by considering a CTC network and an attention decoder. No external language model is involved in this work. The most probable letter sequence $\hat{C}$ given the speech input X is computed as:

$$\hat{C} = \arg\max_{C \in \mathcal{U}^*} \{\lambda \log p_{ctc}(C|X) + (1-\lambda) \log p_{att}(C|X)\} \quad (3)$$

where we can directly estimate the attention posterior distribution $p_{att}(C|X)$ using the chain rule:

$$p_{att}(C|X) = \prod_{l=1}^{L} p(c_l|c_1, ..., c_{l-1}, X). \quad (4)$$

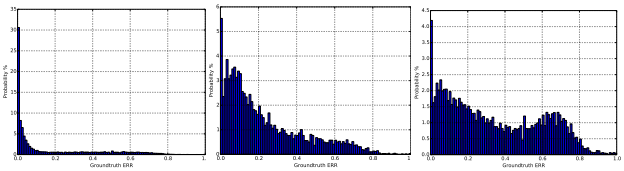
In this work, we apply PM techniques on three kinds of features to predict the CERs: attention distributions $\{a_{lt}\}$, decoder posterior distributions $p(c_l|c_1, ..., c_{l-1}, X)$, and their pre-softmax activations.

3. Data

We conduct all our experiments based on the Wall Street Journal (WSJ) corpus [24] and its variants with additional noises or reverberation conditions. Table 1 summarizes various databases that are used in our experiments. For end-to-end ASR, the clean WSJ SI-284 corpus and Aurora4 multi-condition training data are used in multi-style training. The Aurora4 set [25] contains simulated recordings of WSJ utterances in 14 different acoustic conditions, varying noise types and channel conditions. WSJ dev93 and Aurora4 dev set serve as the validation set for ASR training. We also use the same data configuration to train the auto-encoder. For the RNN predictor, in order to see data with a reasonable balance across different CERs, we use clean WSJ SI-84 together with its two artificially noise-corrupted versions, Aurora4 and CHiME4-Sim. CHiME4-Sim [26] consists of single-channel WSJ data with four additive noises.

Table 1: *Datasets for experiments with CER (%) of Test set*

Task	Dataset	
Train		
ASR	WSJ(SI-284), Aurora4	
AE	WSJ(SI-284), Aurora4	
RNN	WSJ(SI-84), Aurora4, Chime4-Sim	
Dev		
Linear Regr	WSJ, Aurora4, Chime4-Sim	
Test		
All Tasks	WSJ (5.7%)	Aurora4 (14.5%)
	Chime4-Sim (52.2%)	Dirha-Sim (68.2%)
	Chime4-Real (59.0%)	Dirha-Real (70.7%)
	Reverb (41.6%)	



(a) Train (b) Dev (c) Test

Figure 2: *Histogram of CERs in various datasets*

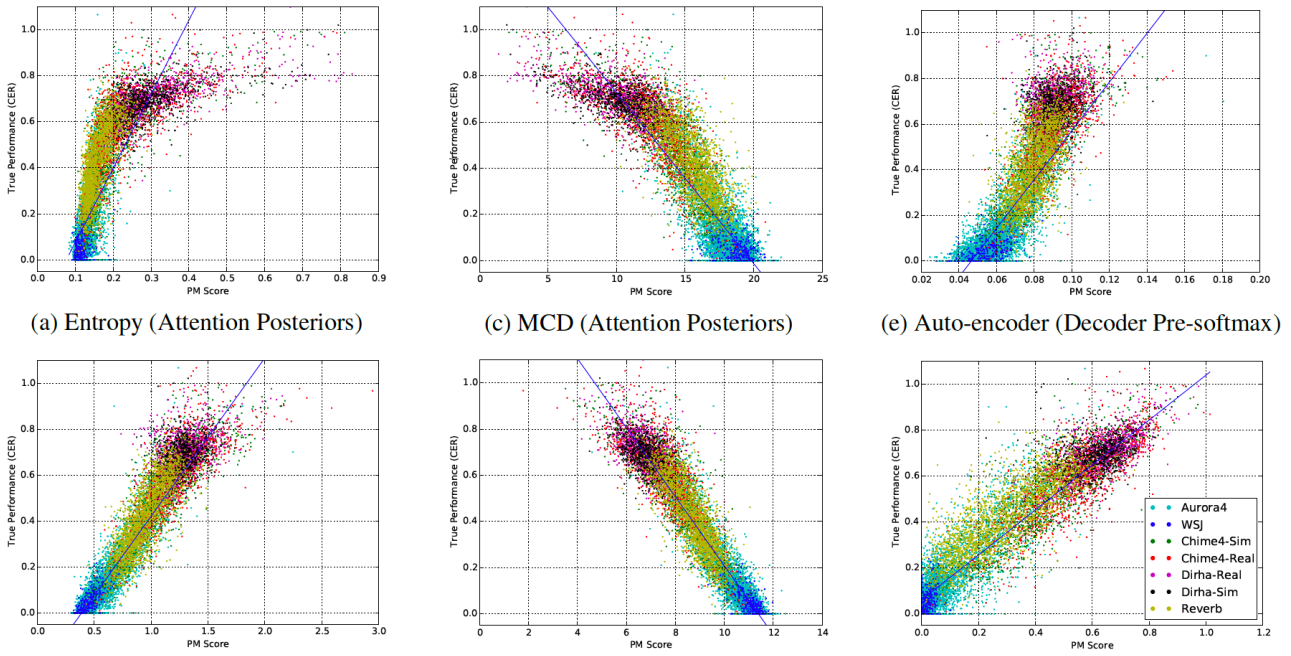
To evaluate the predictability of each PM measure, a linear regression model is applied to map between PM scores and truth performance. The regression model is computed according to development sets from WSJ, Aurora4 and CHiME4-Sim. We refer the data to train the ASR and linear regression model as *Train* and *Dev*, respectively. Fig. 2(a) and Fig. 2(b) show histograms of utterance CERs for both sets. Performance of *Dev* is more widely spread out than *Train*. We test the effectiveness of PM measures with data from two different domains. *Seen* test domains include evaluation sets from WSJ, Aurora4 and CHiME4-Sim which are drawn from the same domains as ASR and PM training; *Unseen* test domains consist of evaluation sets from CHiME4-Real [26] (real noisy recordings), Reverb-Sim [27] (simulated reverberation), Dirha-Sim [28] (simulated reverberation) and Dirha-Real [28] (real reverberated recordings). All the test data together are referenced as *Test*, with utterance CERs shown in Fig. 2(c).

4. Experiments

4.1. Experiment Setup

In the end-to-end model, the encoder contains four BLSTM layers, each with 320 cells in both directions, followed by a 320-unit linear projection layer. A content-based attention mechanism with 320 attention units follows. The decoder is a one-layer unidirectional LSTM with 300 cells. We use 52 distinct labels at the decoder softmax layer, including 26 English letters and additional special tokens (i.e., punctuations and sos/eos).

The model is implemented using the Pytorch backend on ESPnet [29], an end-to-end speech processing toolkit. The model is optimized using the AdaDelta algorithm with a mini-batch size of 15. We also apply a unigram label smoothing technique [30] to avoid over-confident predictions. The beam width is set to 30 for all results. For jointly training with CTC and attention objectives, $\lambda = 0.2$ is used for training, and $\lambda = 0.3$ for decoding. All results are reported as CER. In all



(a) Entropy (Attention Posteriors) (c) MCD (Attention Posteriors) (e) Auto-encoder (Decoder Pre-softmax)

(b) Entropy (Decoder Posteriors) (d) MCD (Decoder Posteriors) (f) RNN Predictor (Decoder Pre-softmax)

Figure 3: *Performance Monitoring Score versus Truth Performance (CER)*

experiments, 80-dimensional mel-scale filter-bank coefficients with additional 3-dimensional pitch features served as the input features. Attention distributions, decoder posteriors, and pre-softmax features are extracted during joint decoding. Decoding results of each *Test* set are shown in Table. 1.

4.2. Entropy

In the hybrid ASR framework, it was observed [3, 19] that the discriminative power of a clean phoneme classifier decreases for input speech with reduced signal-to-noise ratios. As the phoneme posteriors tend to be more uniformly distributed, entropy was proposed as a measure of uncertainty. In end-to-end ASR, which has no phoneme distributions, we investigate if entropy on either the attention probabilities or the decoder posteriors could be a reasonable indicator of model performance.

Entropy is first computed on the character-level distribution $p(c_l|c_1, ..., c_{l-1}, X)$.

$$Entropy(\mathbf{p}) = -\sum_{k=0}^{K} p^{(k)} \log p^{(k)} \tag{5}$$

where $\mathbf{p}_l$ is either the attention probabilities $\mathbf{a}_l$ or decoder posteriors. The utterance-level score is obtained by averaging entropy scores over all predictions.

$$E_{score} = \frac{1}{L} \sum_{l=0}^{L} Entropy(\mathbf{p}_l) \tag{6}$$

Note that the dimension of the attention distribution is equal to the number of time frames T at encoder output, which varies per utterance. So, we normalize this entropy by its upper bound $\log(T)$ for a consistent range $[0, 1]$. Fig. 3(a)(b) show scatter plots of entropy scores versus truth CERs on *Test*, where each point in the plot represents one utterance. A linear model $CER = a * PM + b$ learned to minimize the mean squared error over *Dev* is also shown. Entropy scores on the decoder output clearly demonstrate a linear relationship with true performance, while linear correlation for attention-level distributions holds only for error rates less than 0.5, resulting in larger prediction error overall.

4.3. M-Measure: Mean Character Distance

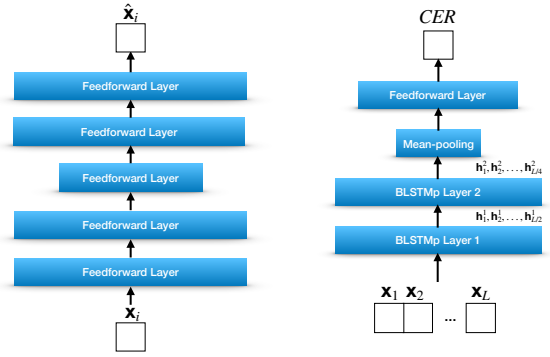
The Mean Temporal Distance (MTD) or M-Measure was proposed to show the mean distance of pair-wise probability distributions from DNN outputs [21]. Symmetric KullbackLeibler divergence was selected as a distance metric for distributions $\mathbf{p}$ and $\mathbf{q}$, which are each posteriors from different time frames.

$$\mathcal{D}(\mathbf{p}, \mathbf{q}) = \sum_{k=0}^{K} p^{(k)} \log \frac{p^{(k)}}{q^{(k)}} + \sum_{k=0}^{K} q^{(k)} \log \frac{q^{(k)}}{p^{(k)}} \tag{7}$$

A high MTD score indicates a greater difference between $\mathbf{p}$ and $\mathbf{q}$, meaning the model is choosing different output classes at different times. In noisy conditions or other cases with low model confidence, the distributions at different times should be more similar. In MTD, M-Measure often needs to sample frame pairs more than 200 ms apart due to phonetic co-articulation; for shorter time spans, small divergence could be caused by high confidence in the same phoneme.

In end-to-end framework, we propose Mean Character Distance (MCD), adapted from mean temporal distance. Since each probability estimate $\mathbf{p}$ in the attention or decoder posterior corresponds to a character prediction, the distance measure is suitable even for adjacent frames without concern for a co-articulation effect. So, we take the mean of distance over all pairs from various windows $\{\triangle l\} = \{1, 2, 3, 4, 5\}$.

$$\mathcal{M}_{score} = \frac{\sum_{\{\triangle l\}} \sum_{l=\triangle l}^{L} \mathcal{D}(\mathbf{p}_{l-\triangle l}, \mathbf{p}_l)}{\prod_{\{\triangle l\}} (L - \triangle l)} \tag{8}$$



(a) Auto-encoder (b) RNN Predictor

Figure 4: *Configurations of two PM techniques*

Equal weights are applied to all pairs, instead of assigning higher weights for more distant pairs, as with MTD. As shown in Fig. 3(c)(d), similar to the observations for entropy measures, PM on decoder posteriors is better for predicting CERs than PM with attention probabilities. Furthermore, MCD is more linearly correlated with CERs than seen with entropy at attention level.

4.4. Auto-Encoder

Mean squared error (MSE) of auto-encoder outputs was proposed in [23] to measure the mismatch between train and test data as an indicator of DNN performance. The auto-encoder is trained to minimize the reconstruction error of the DNN pre-softmax activations from training data. The previous study illustrated that if a data vector is sampled from training data distribution, the corresponding reconstruction error should be low, while a high error could be observed in a mismatched condition.

In the end-to-end network, it is natural to apply this technique to 52-dimensional decoder pre-softmax activations. The auto-encoder used here is a five-layer 512-unit feed-forward neural network, including a 24-dimensional bottleneck layer in the middle as shown in Fig. 4(a). PM score per utterance is derived as average MSE across all frames. In the scatter plot Fig. 3(e), reconstruction error prediction is consistent with the fitted line for CERs lower than 0.4, while the prediction error diverges for utterances with higher CERs. It might be the case that since the auto-encoder only sees the "good" data in training, there is lack of knowledge of how "bad" data looks.

4.5. RNN Predictor

In this work, we propose an RNN-based regression model which directly maps pre-softmax features of one utterance into error rates in the range of CER $[0, +\infty]$. The model is depicted in Fig. 4(b). Two BLSTM layers of 320 units are employed to handle temporal dependencies of inputs. Each layer subsamples every other output frame. A mean-pooling layer is then used on top of BLSTM outputs to formulate one summary vector per utterance, which is fed into a linear layer of 300 units and an output layer with one Rectified Linear unit (ReLu). The model is optimized with MSE loss between predictions and truth CERs. Intuitively, the PM score is derived from the model output. The scatter plot in Fig. 3(f) shows that the predictions are well-aligned with the fitted line from linear regression. Since it is a direct estimation of CER, the ideal fit should be $CER = PM$. The derived model using *Dev* is $CER = 0.98 * PM + 0.06$, which is slightly offset and tilted from the ideal case.

Table 2: *Mean Square Error* ($\times 10^{-2}$) *of linear regression trained on performance monitoring techniques. All results are reported on test sets.*

	Entropy		MCD			
Dataset/Domain	Dec	Att	Dec	Att	AE	RNN
Seen (ASR, PM)						
WSJ	**0.17**	0.88	0.25	0.77	0.82	0.29
Aurora4	**0.44**	1.43	0.47	1.14	1.12	0.74
Seen (PM only)						
CHiME4-Sim	1.13	5.11	1.07	1.87	2.68	**1.01**
Unseen						
CHiME4-Real	1.50	5.77	1.24	2.02	3.62	**1.05**
Reverb-Sim	0.94	3.42	**0.78**	2.20	1.88	1.50
Dirha-Sim	2.23	6.75	**1.16**	2.05	6.07	1.43
Dirha-Real	2.77	11.80	1.26	2.39	6.97	**1.25**
All Together(deg=1)	1.02	3.83	**0.79**	1.67	2.49	1.02
All Together(deg=2)	0.99	2.36	**0.79**	1.71	2.52	0.99
All Together(deg=3)	0.99	1.66	**0.74**	1.63	2.20	0.99
All Together(deg=4)	0.86	2.13	**0.70**	1.33	2.41	0.99
All Together(deg=5)	2.44	5.03	**0.77**	1.35	3.36	0.99

4.6. Overall Results

Table 2 summarizes MSEs of linear regression models trained on various PM techniques across different *Test* sets. It is worth noting that decoder features work more effectively than attention probabilities in all cases for predicting CERs. Entropy gives the lowest MSEs for WSJ and Aurora4, domains which have been seen in ASR training. The RNN predictor achieves best performance in CHiME4-Sim (not surprising, since this domain was seen in RNN predictor training) and real recordings from CHiME4-Real and Dirha-Real, domains not seen in any training stage at all. MCD measure performs the best at the two unseen simulated domains with reverberant conditions. Overall, entropy, MCD, and RNN prediction all provide reasonably good CER predictions, with average prediction errors (square root of MSE) of 10.1%, 8.8% and 10.1%, respectively, where MCD outperforms the rest of PM measures across all test set.

5. Conclusions

In this work, we investigated four different performance monitoring techniques on attention and decoder features from an end-to-end ASR model. Our results show that PM measures on decoder features are more effective for predicting true error rates than PM measures on attention probabilities. Entropy and MCD are very simple, effective measures where MCD shows the overall best performance. And while auto-encoder methods might be suitable to handle mismatch condition within a certain level of data corruption, an RNN-based regression model shows potential in the direction of performance estimation using deep neural network. Overall, these results show great promise for performance prediction of end-to-end ASR models.

6. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1704170 and No. 1743616.

7. References

[1] M. K. Scheffers and M. Coles, "Performance monitoring in a confusing world: Error-related brain activity, judgments of response accuracy, and types of errors." *Journal of Experimental Psychology: Human Perception and Performance*, vol. 26, pp. 141–151, 02 2000.

[2] S. Ganapathy, M. Omar, and J. Pelecanos, "Unsupervised channel adaptation for language identification using co-training," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6857–6861.

[3] S. Okawa, E. Bocchieri, and A. Potamianos, "Multi-band speech recognition in noisy environments," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, vol. 2. IEEE, 1998, pp. 641–644.

[4] S. H. R. Mallidi and H. Hermansky, "A framework for practical multistream asr." in *INTERSPEECH*, 2016, pp. 3474–3478.

[5] A. M. C. Martinez, L. Gerlach, G. P. Vayá, H. Hermansky, J. Ooster, and B. T. Meyer, "Dnn-based performance measures for predicting error rates in automatic speech recognition and optimizing hearing aid parameters," *Speech Communication*, vol. 106, pp. 44–56, 2019. [Online]. Available: https://doi.org/10.1016/j.specom.2018.11.006

[6] X. Wang, R. Li, and H. Hermansky, "Stream attention for distributed multi-microphone speech recognition," *Proc. Interspeech 2018*, pp. 3033–3037, 2018.

[7] X. Wang, R. Li, S. H. Mallid, T. Hori, S. Watanabe, and H. Hermansky, "Stream attention-based multi-array end-to-end speech recognition," *arXiv preprint arXiv:1811.04903*, 2018.

[8] B. T. Meyer, S. H. Mallidi, A. M. C. Martínez, G. Payá-Vayá, H. Kayser, and H. Hermansky, "Performance monitoring for automatic speech recognition in noisy multi-channel environments," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 50–56.

[9] H. Hermansky, "Multistream recognition of speech: Dealing with unknown unknowns," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1076–1088, 2013.

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine learning (ICML)*, 2006, pp. 369–376.

[11] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1764–1772.

[12] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174.

[13] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[14] 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 (NIPS)*, 2015, pp. 577–585.

[15] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4835–4839.

[16] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *INTERSPEECH*, 2017.

[17] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[18] R. Li, X. Wang, S. H. Mallidi, T. Hori, S. Watanabe, and H. Hermansky, "Multi-encoder multi-resolution framework for end-to-end speech recognition," *arXiv preprint arXiv:1811.04897*, 2018.

[19] H. Misra, H. Bourlard, and V. Tyagi, "New entropy based combination rules in hmm/ann multi-stream asr," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 2. IEEE, 2003, pp. II–741.

[20] E. Variani, F. Li, and H. Hermansky, "Multi-stream recognition of noisy speech with performance monitoring." in *Interspeech*, 2013, pp. 2978–2981.

[21] H. Hermansky, E. Variani, and V. Peddinti, "Mean temporal distance: Predicting asr error from temporal properties of speech signal," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7423–7426.

[22] S. H. Mallidi, T. Ogawa, and H. Hermansky, "Uncertainty estimation of dnn classifiers," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 283–288.

[23] S. H. Mallidi, T. Ogawa, K. Veselỳ, P. S. Nidadavolu, and H. Hermansky, "Autoencoder based multi-stream combination for noise robust speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[24] L. D. Consortium, "CSR-II (wsj1) complete," *Linguistic Data Consortium, Philadelphia*, vol. LDC94S13A, 1994.

[25] D. Pearce and J. Picone, "Aurora working group: Dsr front end lvcsr evaluation au/384/02," *Inst. for Signal & Inform. Process., Mississippi State Univ., Tech. Rep*, 2002.

[26] E. Vincent, S. Watanabe, J. Barker, and R. Marxer, "The 4th chime speech separation and recognition challenge," 2016.

[27] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, E. Habets, R. Haeb-Umbach, V. Leutnant, A. Sehr, W. Kellermann, R. Maas, S. Gannot, and B. Raj, "The reverb challenge: A common evaluation framework for dereverberation and recognition of reverberant speech," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct 2013, pp. 1–4.

[28] M. Ravanelli, P. Svaizer, and M. Omologo, "Realistic multi-microphone data simulation for distant speech recognition," in *Interspeech 2016*, 2016.

[29] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "Espnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.

[30] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.