SEQUENCE NOISE INJECTED TRAINING FOR END-TO-END SPEECH RECOGNITION

George Saon, Zoltán Tüske, Kartik Audhkhasi and Brian Kingsbury

IBM Research AI, Yorktown Heights, USA

ABSTRACT

We present a simple noise injection algorithm for training end-to-end ASR models which consists in adding to the spectra of training utterances the scaled spectra of random utterances of comparable length. We conjecture that the sequence information of the "noise" utterances is important and verify this via a contrast experiment where the frames of the utterances to be added are randomly shuffled. Experiments for both CTC and attention-based models show that the proposed scheme results in up to 9% relative word error rate improvements (depending on the model and test set) on the Switchboard 300 hours English conversational telephony database. Additionally, we set a new benchmark for attention-based encoder-decoder models on this corpus.

Index Terms— End-to-end ASR, noise injection

1. INTRODUCTION

The appeal of end-to-end approaches for ASR is a drastic simplification of the pipeline required to build competitive acoustic models. Indeed, all the steps related to discriminative and speaker dependent feature extraction, phonetic context decision trees, HMM alignments, decoding graph construction, etc. can be effectively sidestepped. One may naturally ask the question if DNN-HMM hybrid acoustic models that require detailed knowledge of the aforementioned steps are a thing of the past. The answer, at least insofar as the Switchboard corpus is concerned, is unfortunately negative as demonstrated by the top performing systems [1, 2, 3, 4] which do not include any end-to-end approaches. Consequently, bridging the gap between hybrid and end-to-end models remains an active research area for many sites [5, 6, 7, 8].

It is well known that noise injection in the inputs improves the generalization capability of neural networks [9, 10]. In [10], the author shows that adding Gaussian noise with a small magnitude is equivalent to adding a regularization term to the objective function (squared error or cross-entropy) which encourages the network to converge to a smoother (lower curvature) local optimum. In other words, the training converges to a point where the output of the network is less sensitive to small changes in the input which helps generalization. Noise can be applied at various levels during training such as inputs, weights, or activations (e.g. dropout), but we will limit the discussion to noise applied to the inputs.

Beyond the regularization effect of adding noise, noise benefits in non-linear systems or "stochastic resonance" is a well-studied area of research [11, 12, 13]. In particular, prior works have shown theoretically that addition of carefully-selected noise improves the speed of convergence and performance of the expectation-maximization algorithm [14], backprogation training of feed-forward [15], convolutional [16] and recurrent neural networks [17], Markov chains [18], and threshold neuron signal detection [19].

By their very nature, sequence-to-sequence (S2S) modeling approaches such as connectionist temporal classification [20] and attention-based S2S [21, 22, 23] can benefit greatly from noise injection, perhaps more so than hybrid models. This is because S2S models are much better at memorizing the training data, which is detrimental to their generalization performance. Without noise injection or other forms of data perturbation, the same training utterance is presented to the model over and over again, eventually leading to overfitting. A symptom of this overfitting is a training loss that is significantly smaller than the held-out loss. For example, for the same type of network (6-layer bidirectional LSTM) and same input features, a phone CTC model trained on Switchboard 300 hours has a ratio of 3.6 between held-out and training loss, while in the hybrid HMM case the same size LSTM trained with cross-entropy has a held-out to training loss ratio of only 1.2.

To combat this phenomenon, several authors in the ASR community have looked at noise injection to the inputs during training. In [24], the authors add noisy segments in the time domain to improve the noise robustness of a DNN-HMM hybrid model. In [7], the authors add background noise to 40% of randomly selected utterances in every epoch when training CTC, RNN-T, and attention-based models on a proprietary corpus. Another relevant study is [25], which proposes mixup training [26] in which pairs of input frames from two arbitrary subsequences are interpolated with random weights and the same weights are used for creating soft targets from the corresponding pairs of 1-of-K hard targets (LSTM-HMM hybrid model). The authors show that this helps on a mismatched test set like Hub5'00 CallHome without adversely affecting performance on the matched Hub5'00 SWB test set. Lastly, instead of noise injection, data augmentation techniques which vary the speed and amplitude of the speech signal have also been successful on this task [27].

The paper is organized as follows: in section 2 we study the regularization effect of Gaussian noise on various losses, in section 3 we provide some empirical evidence of the utility of the proposed approach, and in section 4 we summarize our findings.

2. NOISE INJECTION THEORY

2.1. Gaussian noise with cross-entropy loss

Let us first consider the case of Gaussian noise added to the input features and study its effect on the cross-entropy (CE) loss. We will borrow some notations and reproduce the derivation from [10, 24] in the hope that readers who are unfamiliar with it will find the connection between noise injection and CE loss regularization interesting. Denote the set of N input samples $\mathbf{x}_1 \ldots \mathbf{x}_N \in \mathbb{R}^M$ and the corresponding targets $\mathbf{t}_1 \ldots \mathbf{t}_N \in \{0,1\}^K$ where K is the number of classes. Also, let $f(\cdot;\theta) : \mathbb{R}^M \rightarrow \mathbb{R}^K$ be the mapping computed by the neural network parameterized by weights θ. The cross-entropy loss is expressed as:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_i^k \log f_k(\mathbf{x}_i;\theta) \tag{1}$$

with f a continuous and twice-differentiable function of $\mathbf{x}$. Consider $\boldsymbol{\xi}$ a Gaussian noise variable with zero mean and covariance matrix $\epsilon\mathbf{I} \in \mathbb{R}^{M\times M}$ where ϵ is a small positive number. Let us draw N i.i.d. samples $\boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_N$ from $\boldsymbol{\xi}$. Applying the Taylor series expansion to the CE loss viewed as a function of $\boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_N$ around the point $\mathbf{x}_1 \dots \mathbf{x}_N$, the loss with noise injection can be derived as follows:

$$\mathcal{L}(\boldsymbol{\xi};\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_i^k \log f_k(\mathbf{x}_i + \boldsymbol{\xi}_i;\theta) \tag{2}$$

$$\approx \mathcal{L}(\theta) - \sum_{i=1}^{N}\sum_{k=1}^{K} t_i^k \left[{\boldsymbol{\xi}_i}^T \frac{\nabla f_k(\mathbf{x}_i)}{f_k(\mathbf{x}_i)} + \frac{1}{2}{\boldsymbol{\xi}_i}^T \mathbf{H}_k(\mathbf{x}_i)\boldsymbol{\xi}_i \right] \tag{3}$$

where we have dropped the dependency of f_k on θ for easier readability. The second order terms $\mathbf{H}_k$ are defined as

$$\mathbf{H}_k(\mathbf{x}) = -\frac{1}{f_k^2(\mathbf{x})}\nabla f_k(\mathbf{x})\nabla f_k(\mathbf{x})^T + \frac{1}{f_k(\mathbf{x})}\nabla^2 f_k(\mathbf{x}) \tag{4}$$

Since $\boldsymbol{\xi}_i$ is independent of $\frac{\nabla f_k(\mathbf{x}_i)}{f_k(\mathbf{x}_i)}$ and $\mathbb{E}\{\boldsymbol{\xi}\} = 0$, the first-order term vanishes and we are left with:

$$\mathcal{L}(\boldsymbol{\xi};\theta) \approx \mathcal{L}(\theta) - \frac{\epsilon}{2}\mathrm{Tr}\left\{ \sum_{k=1}^{K} \sum_{i=1 \ s.t.\ t_i^k=1}^{N} \mathbf{H}_k(\mathbf{x}_i) \right\} \tag{5}$$

The last equation shows that noise injection training is equivalent to placing a regularization term on the loss with the weight controlled by ϵ, the magnitude of the noise. According to [10], $\mathbf{H}_k(\mathbf{x})$ simplifies to $-\frac{1}{f_k^2(\mathbf{x})}\nabla f_k(\mathbf{x})\nabla f_k(\mathbf{x})^T$ which is negative semi-definite, meaning that the new loss function $\mathcal{L}(\boldsymbol{\xi};\theta)$ will accept solutions in regions with low curvature. This makes the loss function (and the network output) less sensitive to small changes in the inputs which leads to better generalization.

2.2. Gaussian noise with CTC and attention loss

Here we extend the previous arguments to the CTC and attention losses. Let $\mathbf{y} = y_1 \dots y_L,\ y_i \in \{1,\dots,K\}$ be a target sequence of length L composed of either phones, characters or words and $\mathbf{x} = \mathbf{x}_1 \dots \mathbf{x}_T$ the corresponding M-dimensional feature vector sequence with length $T \geq L$. The CTC loss can be expressed as:

$$\mathcal{L}(\theta) = -\log p(\mathbf{y}|\mathbf{x}) \tag{6}$$

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z}\in\Phi^{-1}(\mathbf{y})} p(\mathbf{z}|\mathbf{x}) = \sum_{\mathbf{z}\in\Phi^{-1}(\mathbf{y})} \prod_{t=1}^{T} f_{z_t}(\mathbf{x};\theta) \tag{7}$$

where $\mathbf{z} = z_1 \dots z_T,\ z_i \in \{0,\dots,K\}$ are alignments consistent with $\mathbf{y}$ obtained by either repeating symbols from $\mathbf{y}$ or by inserting special BLANK symbols. $\mathbf{z} \overset{\Phi}{\mapsto} \mathbf{y}$ is a mapping which removes BLANKs and duplicate symbols from $\mathbf{z}$ and f is the $(K+1)$-dimensional mapping computed by the neural network with weights θ. Note that here, unlike for the cross-entropy case, the outputs at time t depend on the entire input sequence $\mathbf{x}$ not only on $\mathbf{x}_t$. The sum of products in (7) can be efficiently calculated by a forward-backward algorithm with a $(2L+1)$-state HMM topology with skippable BLANK states as shown in [20].

In the case of attention loss, the posterior probability of the output sequence $\mathbf{y} = y_1 \dots y_L$ given the input sequence $\mathbf{x} = \mathbf{x}_1 \dots \mathbf{x}_T$ can be written as:

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{h}) = \prod_{l=1}^{L} p(y_l|\mathbf{c}_l, y_1, \dots, y_{l-1}) \tag{8}$$

where $\mathbf{c}_l$ is the context for decoding step l and is computed by weighting the encoder hidden state sequence $\mathbf{h} = \mathbf{h}_1 \dots \mathbf{h}_M$ ($M \leq T$ due to subsampling) by the attention γ i.e.:

$$\mathbf{c}_l = \sum_{t=1}^{M} \gamma_{l,t}\mathbf{h}_t \tag{9}$$

$$\gamma_{l,t} = \exp(e_{l,t}) / \sum_{t'=1}^{M} \exp(e_{l,t'}) \tag{10}$$

$$\mathbf{e}_l = f(\mathbf{h}, \gamma_{l-1}, \mathbf{g}_{l-1}) \tag{11}$$

where $\mathbf{g}_l$ is the hidden state of the decoder at step l and $\mathbf{e}_l$ can be computed in different ways depending on the attention mechanism.

Like in the previous subsection, define the noise injected loss for either CTC or attention as:

$$\mathcal{L}(\boldsymbol{\xi};\theta) = -\log p(\mathbf{y}|\mathbf{x} + \boldsymbol{\xi}) \tag{12}$$

with $\boldsymbol{\xi} = \boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_T$ i.i.d. samples drawn from $\mathcal{N}(\mathbf{0}, \epsilon\mathbf{I})$. Using the property $\mathbb{E}\{XY\} = \mathbb{E}\{X\}\mathbb{E}\{Y\}$ for independent r.v.'s X and Y, it can be shown that, like for the cross-entropy loss, the first-order term in the Taylor expansion of $\mathcal{L}(\boldsymbol{\xi};\theta)$ around $\mathbf{x}$ is zero and we are left with:

$$\mathcal{L}(\boldsymbol{\xi};\theta) \approx \mathcal{L}(\theta) + \frac{\epsilon}{2}\mathrm{Tr}\left\{ \frac{\partial^2 \mathcal{L}(\boldsymbol{\xi};\theta)}{\partial \boldsymbol{\xi}^2}\Big|_{\boldsymbol{\xi}=\mathbf{0}} \right\} \tag{13}$$

where the second-order term is positive definite. Hence, the noise-injected CTC and attention losses will have local minima in regions of low curvature as well.

3. EXPERIMENTS AND RESULTS

All experiments were carried out on the Switchboard English conversational telephony corpus. The training set for our acoustic models consists of 262 hours of Switchboard 1 audio with segmentations and transcripts provided by Mississippi State University which, in keeping with tradition, we refer to as the Switchboard 300 hours corpus. Similar to what we did in [3], we report results not only on the Hub5'00 Switchboard (SWB) and CallHome (CH) test sets, but also on RT'02 (6.4h, 120 speakers, 64K words), RT'03 (7.2h, 144 speakers, 76K words) and RT'04 (3.4h, 72 speakers, 36.7K words). This is done in order to avoid overfitting to the Hub5'00 test sets, which we consider to be stale due to extensive use by us and other sites for almost two decades.

3.1. Phone CTC experiments

The models are trained on VTL-warped log-mel spectra extracted every 10ms using a Mel filterbank with 40 frequency bins. The features are mean and variance normalized per conversation side and augmented with Δ and $\Delta\Delta$ coefficients. We also skip every other frame and stack every two consecutive frames to lessen the computational demands on the neural network. Lastly, we append a 100-dimensional speaker-dependent i-vector [28] resulting in 50 340-dimensional feature vectors per second.

We have trained 6-layer bidirectional LSTMs with 1024 cells per layer and an output layer with 44 phones plus the BLANK symbol using CTC with Nesterov momentum and a dropout rate of 0.25 for 19 epochs. We have used a fixed learning rate schedule where the learning rate is kept constant at 0.01 for the first 10 epochs then gets annealed by $1/\sqrt{2}$ every epoch after that. Utterances are sorted in ascending length order and are grouped into batches of size 32. The batches are traversed in increasing length for the first epoch and are randomly shuffled for the remaining epochs. The implementation is done in PyTorch [29] with NVIDIA's cuDNN backend and Baidu's `warp-ctc` extension for computing the CTC loss and gradients. The training time is approximately one day on one P100 GPU. We have experimented with the following types of noises applied to an utterance $\mathbf{x}_1 \dots \mathbf{x}_T$:

- Gaussian noise (GN): $\mathbf{x}'_i = \mathbf{x}_i + \boldsymbol{\xi}_i,\ \boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$,
- Sequence noise (SN): $\mathbf{x}'_i = \log[\exp(\mathbf{x}_i) + \lambda \exp(\mathbf{n}_i)]$ for a random utterance $\mathbf{n}_1 \dots \mathbf{n}_T$ and
- Sequence noise with randomized frames (RF): $\mathbf{x}'_i = \log[\exp(\mathbf{x}_i) + \lambda \exp(\mathbf{n}_{\tau(i)})]$, where τ is a random permutation of $\{1, \dots, T\}$.

For the Gaussian noise experiments, we tried standard deviations in the range 0.1 to 0.5 whereas for the sequence noise experiments, we varied λ also from 0.1 to 0.5 and selected a random subset of 20% of utterances without noise at every epoch. Note that, for SN and RF, the noise is only added to the spectra and the Δ, $\Delta\Delta$ coefficients are recomputed whereas for GN, the noise is added to all dimensions.

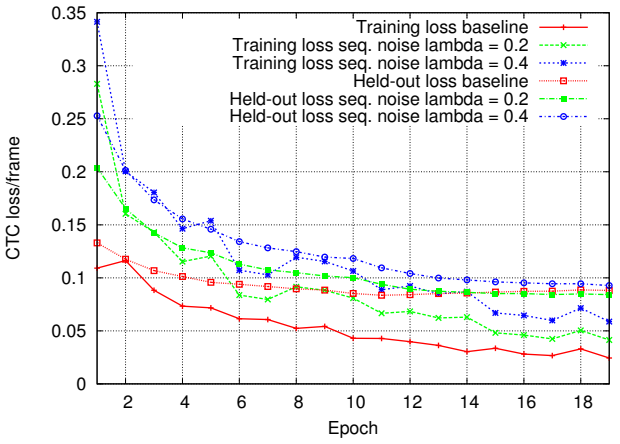

Fig. 1. Training and held-out phone CTC losses for baseline and sequence noise experiments.

In Figure 1, we show the training and held-out phone CTC losses for the sequence noise experiments with two values of λ. We observe that, for larger λ, the training and held-out losses are closer to each other. Also, the held-out losses converge to similar values whereas the training losses are significantly different for the three runs.

	SWB	CH	RT02	RT03	RT04	Avg.
Baseline	11.6	21.1	20.1	19.3	17.9	18.0
GN $\sigma = 0.1$	11.7	20.9	19.8	18.9	17.9	17.8
GN $\sigma = 0.2$	11.5	20.8	19.3	18.5	17.9	17.6
GN $\sigma = 0.4$	11.7	20.6	19.3	18.4	17.4	17.5
GN $\sigma = 0.5$	11.9	20.9	19.2	**18.2**	17.5	17.5
SN $\lambda = 0.1$	11.3	20.6	18.9	18.8	17.2	17.4
SN $\lambda = 0.2$	**10.9**	20.7	18.7	18.6	16.9	17.2
SN $\lambda = 0.4$	11.0	**20.5**	**18.3**	18.4	**16.8**	**17.0**
SN $\lambda = 0.5$	11.4	21.1	18.6	18.8	17.0	17.4
RF $\lambda = 0.4$	11.3	21.0	18.7	18.9	17.6	17.5

Table 1. Word error rates for the different noise injection experiments for phone CTC models trained on Switchboard 300 hours.

All decodings were done using our legacy 30K word vocabulary and a 4-gram language model with 4M n-grams. Table 1 shows the recognition results for the various experiments. We note that all three noise addition schemes result in gains over the baseline with an edge for the sequence noise without frame randomization where the gains range from 3% relative (CH) to 9% relative (RT'02) with an average of 6% relative WER improvement. Frame randomization in the noise utterances decreases the gains which shows that it is important to preserve the sequence information in the noise.

3.1.1. Switchboard 2000 hours experiment

We wanted to verify whether the proposed technique scales up with more training data by training phone CTC LSTMs on the Switchboard + Fisher 2000 hours set. The models have the same input features and architectures as the SWB 300h LSTMs, the only difference being that the output layer has 89 units corresponding to 88 word position-dependent phones + BLANK. The initial learning rate was set to 0.005 and gets annealed by $1/\sqrt{2}$ every epoch after epoch number 8. The training time for 19 epochs was roughly 6 days on one V100 GPU. For sequence noise injection, we used the best configuration from our SWB 300h setup. Decoding was done with an 85K word vocabulary and a 4-gram language model with 36M n-grams [3]. The results are shown in Table 2.

	SWB	CH	RT02	RT03	RT04	Avg.
Baseline	**8.0**	13.9	12.4	11.7	10.5	11.3
SN $\lambda = 0.4$	8.0	**13.2**	**11.8**	**11.1**	**10.2**	**10.9**

Table 2. Word error rates for phone CTC models with sequence noise injection trained on Switchboard+Fisher 2000 hours.

Unsurprisingly, the gains (3.5% relative) are less compared to the SWB 300 hours case but still substantial across all test sets except for Hub5'00 SWB. Interestingly, the improvements on the Hub5'00 CallHome testset are *larger* than for the 300 hours training scenario.

3.2. Attention based encoder-decoder experiments

Our attention based encoder-decoder model is a modified version of [30], and has the following structure. The encoder consists of 6

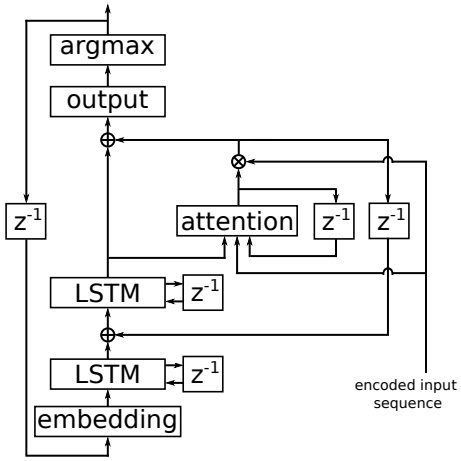

Fig. 2. The attention based decoder network used in the encoder-decoder experiments.

bidirectional LSTM layers, each having 384 nodes. Its input is 40-dimensional vocal tract length normalized MFCC. The first layers apply pyramidal processing as [21]. The input frame rate is reduced by four in total, twice by factor of two at the outputs of the first and second LSTM layers using max pooling. The LSTM outputs from the forward and backward directions are summed up after every layer. The final dimension of the encoder output is 256, enforced by a linear bottleneck.

The decoder network is summarized in Fig. 2. As can be seen, it contains two unidirectional LSTM layers with 256 nodes each, and a single attention mechanism. A dedicated LSTM layer on top of the embedding corresponds to a language-model-like component because its output depends only on the history of the decoded symbols. The decoder network uses location aware attention mechanism as in [23]. The previous attention is mapped to a 256-channel signal by 1-D convolution using 5-dimensional kernels. The energy function of the attention calculation is performed by a single layer rectified-linear-unit (ReLU) feed-forward neural network. It processes the sum of the encoder output, the smoothed attention, and the output of the decoder LSTM. Thus, all these three components have a fixed size of 256. The output (and the embedding) models either 42 character or 600 sub-word units generated by byte-pair encoding (BPE) [31].

All models were trained on the Switchboard 300 hours set by optimizing cross-entropy loss only. The baseline character model was trained from scratch for 60 epochs with a batch size of 32. Next, this seed model was fine-tuned towards the final output targets using a batch size of 16 for another 30 epochs. The model training was carried out by teacher forcing with 80% probability [32]. Furthermore, a 20% dropout rate was applied to the LSTM outputs in the encoder whereas in the decoder, the embedding output was dropped with 5% and the LSTM output with 10% probability. In the initial epochs, sequences were presented in increasing length order to the network, similar to the CTC experiments.

The final performance of the encoder-decoder network was optimized on the Swichboard (SWB) part of the Hub5'00 set by tuning the following hyper-parameters: length normalization, coverage penalty term, posterior smoothing, or optionally the interpolation weight of the language model whose score was fused in a shallow fashion. The search used a fixed beam size of 50 and was constrained by a lexical prefix tree. The neural network language models (NNLM) were trained on 24M words of the joint set of Switchboard 300 and Fisher data. The NNLMs have an embedding size of 512, two LSTM layers with 2048 nodes, and a 128-dimensional bottleneck layer before the softmax output.

The effect of sequence noise injection on our attention based encoder-decoder model is summarized in Table 3. We obtained the best results when the sequence noise of a random utterance was injected with 40% probability at $\lambda = 0.3$ scale. As shown, sub-word units performed slightly better – 0.3% on average – than the character model, but not on the development set (SWB). The application of sequence noise on top of the BPE model resulted in a 0.7% absolute gain. The positive effect of noise injection is clearly visible even after decoding with the neural network language model and, to the best of our knowledge, we have achieved a new state-of-the-art recognition performance with attention-based models on the Switchboard 300 hours corpus.

LM	Output	SWB	CH	RT02	RT03	RT04	Avg.
	char	12.7	24.3	22.5	23.1	22.5	21.0
	BPE 600	12.7	23.2	21.8	22.2	23.7	20.7
	+seq. noise	**11.8**	**22.6**	**21.1**	**21.9**	**22.5**	**20.0**
×	BPE 600	10.6	22.1	19.2	20.2	20.7	18.6
	+seq. noise	**10.0**	**21.7**	**18.5**	**20.1**	**19.9**	**18.0**

Table 3. Effect of sequence noise injection on various attention based encoder-decoder models trained on Switchboard 300 hours.

4. DISCUSSION

Sequence noise injection improves the performance of end-to-end models more than adding simple Gaussian noise or speech frames without sequence information. We surmise that the structure in the noise helps to enhance (or suppress) various regions in the input signal more effectively than noise frames without structure. What is missing from this work is a theoretical insight beyond the Gaussian case on why sequence information in the noise is important. Also, from an empirical point of view, it is unclear if gains from data augmentation and sequence noise injection are additive and we plan to address that in future experiments.

5. REFERENCES

[1] Kyu J Han, Akshay Chandrashekaran, Jungsuk Kim, and Ian Lane, "The CAPIO 2017 conversational speech recognition system," *arXiv preprint arXiv:1801.00059*, 2017.

[2] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke, "The Microsoft 2017 conversational speech recognition system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[3] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al., "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[4] William Hartmann, Roger Hsiao, Tim Ng, Jeff Ma, Francis Keith, and Man-Hung Siu, "Improved single system conversational telephone speech recognition with VGG bottleneck features," *Proc. Interspeech 2017*, pp. 112–116, 2017.

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

[6] Hagen Soltau, Hank Liao, and Hasim Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," *arXiv preprint arXiv:1610.09975*, 2016.

[7] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," in *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017, pp. 206–213.

[8] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny, "Building competitive direct acoustics-to-word models for English conversational speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4759–4763.

[9] Kiyotoshi Matsuoka, "Noise injection into inputs in backpropagation learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 436–440, 1992.

[10] Chris M Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural computation*, vol. 7, no. 1, pp. 108–116, 1995.

[11] Adi R Bulsara and Luca Gammaitoni, "Tuning in to noise," *Physics today*, vol. 49, no. 3, pp. 39–47, 1996.

[12] Luca Gammaitoni, Peter Hänggi, Peter Jung, and Fabio Marchesoni, "Stochastic resonance," *Reviews of modern physics*, vol. 70, no. 1, pp. 223, 1998.

[13] Bart Kosko, *Noise*, Penguin, 2006.

[14] Osonde Osoba, Sanya Mitaim, and Bart Kosko, "The noisy expectation–maximization algorithm," *Fluctuation and Noise Letters*, vol. 12, no. 03, pp. 1350012, 2013.

[15] Kartik Audhkhasi, Osonde Osoba, and Bart Kosko, "Noise benefits in backpropagation and deep bidirectional pre-training," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, 2013, pp. 1–8.

[16] Kartik Audhkhasi, Osonde Osoba, and Bart Kosko, "Noise-enhanced convolutional neural networks," *Neural Networks*, vol. 78, pp. 15–23, 2016.

[17] Olaoluwa Adigun and Bart Kosko, "Using noise to speed up video classification with recurrent backpropagation," in *Neural Networks (IJCNN), 2017 International Joint Conference on*, 2017, pp. 108–115.

[18] Brandon Franzke and Bart Kosko, "Noise can speed convergence in Markov chains," *Physical Review E*, vol. 84, no. 4, pp. 041112, 2011.

[19] Ashok Patel and Bart Kosko, "Error-probability noise benefits in threshold neural signal detection," *Neural Networks*, vol. 22, no. 5-6, pp. 697–706, 2009.

[20] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 369–376.

[21] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4960–4964.

[22] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[23] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, 2015, pp. 577–585.

[24] Shi Yin, Chao Liu, Zhiyong Zhang, Yiye Lin, Dong Wang, Javier Tejedor, Thomas Fang Zheng, and Yinguo Li, "Noisy training for deep neural networks in speech recognition," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, pp. 2, 2015.

[25] Ivan Medennikov, Yuri Khokhlov, Aleksei Romanenko, Dmitry Popov, Natalia Tomashenko, Ivan Sorokin, and Alexander Zatvornitskiy, "An investigation of mixup training strategies for acoustic models in ASR," in *Interspeech 2018*. ISCA, 2018.

[26] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[27] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "Audio augmentation for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[28] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *Proc. ASRU*, 2013.

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[30] "https://github.com/awni/speech.git," .

[31] "https://github.com/rsennrich/subword-nmt," .

[32] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.