Self-Attentional Acoustic Models

Matthias Sperber[1], *Jan Niehues*[1], *Graham Neubig*[2], *Sebastian Stüker*[1], *Alex Waibel*[12]

[1]Karlsruhe Institute of Technology
[2]Carnegie Mellon University
{first}.{last}@kit.edu, gneubig@cs.cmu.edu

Abstract

Self-attention is a method of encoding sequences of vectors by relating these vectors to each-other based on pairwise similarities. These models have recently shown promising results for modeling discrete sequences, but they are non-trivial to apply to acoustic modeling due to computational and modeling issues. In this paper, we apply self-attention to acoustic modeling, proposing several improvements to mitigate these issues: First, self-attention memory grows quadratically in the sequence length, which we address through a downsampling technique. Second, we find that previous approaches to incorporate position information into the model are unsuitable and explore other representations and hybrid models to this end. Third, to stress the importance of local context in the acoustic signal, we propose a Gaussian biasing approach that allows explicit control over the context range. Experiments find that our model approaches a strong baseline based on LSTMs with network-in-network connections while being much faster to compute. Besides speed, we find that interpretability is a strength of self-attentional acoustic models, and demonstrate that self-attention heads learn a linguistically plausible division of labor.[1]

Index Terms: speech recognition, acoustic model, self-attention

1. Introduction

In order to transform an acoustic signal into a useful abstract representation, acoustic models must take into account the complex interplay of local and global dependencies in an acoustic signal. At a local, temporally constrained level, we observe concrete linguistic events (phonemes), while at a global level the signal is influenced by factors such as channel and voice properties. Traditional acoustic models reflect this intuition about global and local dependencies by first applying a normalization phase, a global operation that aims at producing invariance with respect to channel and speaker characteristics. After this, traditionally a hidden Markov model is applied over polyphones, modeling only local dependencies (beads-on-a-string view [1]). This restriction has in part been motivated by the intuition that global effects should be removed from the signal at this stage.

However, the empirical success of recurrent neural networks (RNNs) for acoustic modeling [2] has challenged this intuition and indicated that considering the global context is still beneficial at this stage. Unfortunately, RNNs suffer from slow computation speed and may not be able to optimally exploit long-range context. Self-attentional architectures [3, 4, 5] have recently shown promising results as an alternative to RNNs for modeling discrete sequences [6]. These models relate different positions in a sequence by computing pairwise similarities, in order to compute a higher level representation of the sequence. Self-attention is attractive (1) computationally because it can be efficiently implemented through batched tensor multiplication, and (2) from a modeling perspective because it allows direct conditioning on both short range-context and long-range context, without the need to pass information through many intermediate states as is the case with RNNs.

In this paper, we explore self-attentional architectures for acoustic modeling, by using the listen-attend-spell model [7] and replacing its pyramidal encoder component with self-attention.[2] In order to make self-attentional architectures work for acoustic modeling, several challenges must be addressed. First, self-attention computes the similarity of each pair of inputs, so the amount of memory grows quadratically with respect to the sequence length. This is problematic for modeling acoustic sequences, because these can get very long, e.g. our training utterances contain up to 2026 frames (800 on average). To address this issue, we apply downsampling by reshaping the sequence before self-attentional layers.

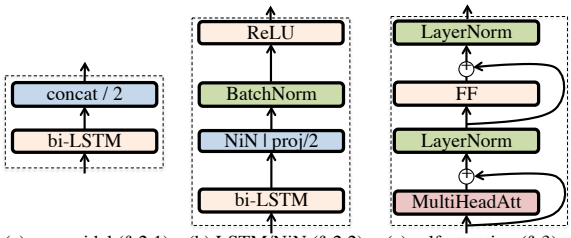
The second challenge is incorporating positional information into the model. Unlike an RNN, self-attention has no inherent mechanism of modeling sequence position. Vaswani et al. [6] propose an additive trigonometric position encoding, which is problematic in the case of acoustic modeling because our inputs are fixed speech features rather than flexibly learned word embeddings. While concatenating positional embeddings instead provides some remedy, we find it necessary to design a hybrid self-attention/RNN architecture to obtain good results.

The third challenge is effective modeling of context relevance. Speech frames contain much less information than words and it is therefore more difficult to estimate the importance of pairs of frames with respect to each other. Based on the intuition that locality of context plays a special role in acoustic modeling, we propose to apply diagonal Gaussian masks with learnable variance to attention heads. This gives attention heads more control over context relevance and improves word error rates consistently, by up to 1.59%. We observe that while bottom layer attention heads converge toward diversity in context range, higher layers use long-range context.

Attention mechanisms improve the often criticized poor interpretability of neural end-to-end models because they enforce an explicit expression of dependencies. Self-attention brings this interpretability inside the encoder, making it possible to examine how speech is encoded before making the final recognition decisions. Our analysis reveals that different attention heads measure similarity along different linguistically plausible dimensions such as phoneme clusters, indicating that they function in part to reduce acoustic variability by establishing averaged versions of matching acoustic events across the utterance.

[1]Code at http://msperber.com/research/self-att

[2]We also refer to independent work that has concurrently addressed similar questions [8].



(a) pyramidal (§ 2.1) (b) LSTM/NiN (§ 2.2) (c) self-attention (§ 3)

Figure 1: *Block diagrams of baselines and the core model.*

2. Attentional Models for ASR

2.1. Listen, Attend, Spell

Our ASR model is based on the listen-attend-spell model [**?**], an attentional encoder-decoder model [9, 10, 11] where the encoder component serves as an acoustic model, taking speech features as input. Because acoustic sequences are very long, the encoder performs downsampling to make memory and runtime manageable. This is achieved through a pyramidal LSTM (Fig. 1a), a stack of LSTM layers where pairs of consecutive outputs of a layer are concatenated before being fed to the next layer, such that the number of states is halved between layers.

2.2. Existing Encoders for Speech

Several improvements over the pyramidal LSTM encoder have been proposed [12]. As a second baseline, we employ a state-of-the-art model [12] that stacks blocks consisting of an LSTM, a network-in-network (NiN) projection, and batch normalization (Fig. 1b). The top LSTM/NiN block is extended by a final LSTM layer. NiN denotes a simple linear projection applied at every time step, possibly performing downsampling by concatenating pairs of adjacent projection inputs.

3. Self-Attentional Acoustic Models

Self-attention is applied to a sequence of state vectors and transforms each state into a weighted average over all the states in the sequence, with more relevant states being given more influence. The underlying intuition is that states at each time step should be conditioned on the most relevant states across the whole sequence. Our basic form of self-attention follows Vaswani et al. [6], where relevance is measured by computing dot product similarity after applying a linear projection to both vectors. For acoustic sequences, neighboring frames are naturally similar if they represent parts of the same acoustic event. When an event with similar acoustic characteristics appears at different places in an utterance, those occurrences would be deemed relevant, as well. Following [6] we use 8 attention heads where each head can compute this similarity independently.

Our model is specifically computed as follows (Fig. 1c):

$$Q_i{=}XW_i^{\mathrm{Q}}, K_i{=}XW_i^{\mathrm{K}}, V_i{=}XW_i^{\mathrm{V}} \quad (1)$$

$$\text{head}_i = \mathrm{softmax}(\frac{Q_i K_i^{\mathrm{T}}}{\sqrt{d}})V_i \quad \forall i \quad (2)$$

$$\text{MultiHeadAtt} = \mathrm{concat}(\text{head}_1, \text{head}_2, \ldots) \quad (3)$$

$$\text{MidLayer} = \mathrm{LayerNorm}\left[\text{MultiHeadAtt} + X\right] \quad (4)$$

$$\text{SAL} = \mathrm{LayerNorm}\left[\mathrm{FF}\left(\text{MidLayer}\right) + \text{MidLayer}\right] \quad (5)$$

Here, $X{\in}\mathbb{R}^{l\times d}, Q_i, K_i, V_i{\in}\mathbb{R}^{l\times d/n}$ denote inputs and their query/key/value transformations for attention heads indexed by $i{\in}\{1,\cdots,8\}$, sequence length l, and hidden dimension d. SAL denotes the final output of the self attention layer. $W_i^{\mathrm{Q}}, W_i^{\mathrm{K}}, W_i^{\mathrm{V}} \in \mathbb{R}^{d\times d/n}$ are parameter matrices. FF is a position-wise feed-forward network intended to introduce additional depth and nonlinearities, defined as $\mathrm{FF}(x){=}\max\left(0, xW_1 + b_1\right)W_2 + b_2$. LayerNorm is according to [13].

4. Tailoring Self-Attention to Speech

4.1. Downsampling

To introduce downsampling so that the model described in § 3 fits in memory, we apply a reshaping operation before every self-attention block. This reduces the sequence length by a factor a and increases the vector state dimension accordingly:

$$X \in \mathbb{R}^{l\times d} \rightarrow_{\text{reshape}} \hat{X} \in \mathbb{R}^{\frac{L}{a}\times ad}$$

We then compute (1) through (5) as before, with the shape of weight matrices adjusted to $W_i^{\mathrm{Q}}, W_i^{\mathrm{K}}, W_i^{\mathrm{V}} \in \mathbb{R}^{da\times d/n}$. This reduces the memory consumption of the attention matrix by factor a^2. It is crucial to apply reshapes also before the bottom layer so that the large bottom attention matrix is scaled down. Note that this approach is very similar to downsampling as in the pyramidal LSTM, except that it is applied to a sequence feature matrix instead of per-timestep, and also applied before the bottom layer.

4.2. Position Modeling

Position information is crucial in sequence-to-sequence models, but self-attention is completely agnostic to sequence positions. Prior works added trigonometric position encodings [6] or learned position embeddings [14] to input vectors, but we found that this approach does not work well for acoustic sequences. This is intuitive, as the inputs are fixed feature vectors rather than trainable word embeddings, making it difficult for the model to separate position and content for each state.

4.2.1. Concatenated Position Representation

A straight-forward solution to enable separation of position and content for fixed inputs is to concatenate position representation instead of using a sum. We explore three variants: First, concatenating trigonometric encodings [6] to the input feature vectors. Second, concatenating learned embeddings [14] to inputs. Third, concatenating separately learned position embeddings to the queries and keys (Q,K in Equation 1) so that the key and query position can be taken into account when computing relevance at each layer.

4.2.2. Hybrid Models

RNNs are effective at keeping track of positional information. We can exploit this by introducing recurrent layers into our encoder. We explore two alternatives:

Stacked hybrid model. Here, we stack 2 LSTM/NiN blocks (Fig. 1b) without downsampling, followed by a final LSTM, on top of our self-attention layers. This approach does not make the self-attention layers themselves position-aware, but the final encoder states are position-aware. Reversing the order of self-attention and LSTM/NiN is also conceivable but would compromise speed because slow recurrent computations are applied before downsampling.

Interleaved hybrid model. Another option is to replace the feed-forward operation (FF in Equation 5) by an LSTM. Note that this introduces LSTMs before the sequence is fully downsampled and therefore compromises some of the speed gains. On the other hand, it allows the higher self-attention layers to take advantage of position information encoded by lower interleaved LSTMs.

4.3. Attention Biasing

Self-attention allows direct conditioning on the whole sequence, but it is unclear to what extent this is beneficial for our acoustic model. While context required to model polyphones may span only a relatively small temporal window, remaining channel and speaker properties may require long-range context. To account for the special role of context locality in acoustic modeling, we introduce an explicit way of controlling the context range by using a bias matrix $M \in \mathbb{R}^{l \times l}$ and computing $\text{head}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d}} + M)V_i$. By setting values around the diagonal of this mask to a higher value, we can bias the self-attention toward attending in a local range around each frame.

4.3.1. Local Masking

We can apply hard masking by setting M as an inversely banded matrix of bandwith $b \in \mathbb{N}_{\text{odd}}$ with

$$M_{jk} = \begin{cases} 0 & |j-k| < \frac{b}{2} \\ \text{-}\infty & \text{else} \end{cases}.$$

As a result, all attention weights outside the band are set to 0, so that the self-attention is restricted to a local region of size b. The hyperparameter b can be set prior to training such that the model effectively attends to a range similar to polyphone context in hidden Markov models. Notice the similar idea explored concurrently in independent work [8].

4.3.2. Gaussian Bias

For more flexibility, we use a soft Gaussian mask by defining

$$M_{jk} = \frac{-(j-k)^2}{2\sigma^2}.$$

σ is a trainable standard deviation parameter. It is learned separately for each attention head so that the context range can differ between attention heads. Besides more modeling expressiveness, the learned variances can also be inspected and may help us to understand and interpret the model.[3] Note that this bears some resemblance to prior work [16] who use a *linear* distance map instead of a Gaussian and do not include trainable parameters, making their model less flexible and less interpretable.

5. Experimental Setup

We focus our experiments on the TEDLIUM corpus [17], a widely used corpus of 200h of recorded TED talks, with the development split used as validation data. Our implementation is based on the XNMT toolkit, with which we have previously demonstrated competitive ASR results on two benchmarks [18].

The training settings follow [18] where relevant. We extract 40-dimensional Mel filterbank features with per-speaker mean and variance normalization using Kaldi [19]. We exclude utterances longer than 1500 frames to keep memory requirements manageable. The encoder-decoder attention is MLP-based, and the decoder uses a single LSTM layer. The number of hidden units is 128 for the encoder-decoder attention MLP, 64 for target character embeddings, and 512 elsewhere unless otherwise noted. The model uses input feeding [20], variational recurrent dropout with probability 0.2 and target character dropout with probability 0.1 [21]. We apply label smoothing [22] and fix the target embedding norm to 1 [23]. For inference, we use a beam size of 20 and length normalization with exponent 1.5. Self-attention layers use a hidden dimension of 256 and feed-forward dimension of 256, and attention dropout with probability 0.2. When LSTMs are part of the encoder, we use bidirectional LSTMs with 256 hidden units per direction. Concatenated position representation vectors are of size 40.

The vocabulary consists of the 26 English characters, apostrophe, whitespace, and special start-of-sequence and unknown-character tokens. We set the batch size dynamically depending on the input sequence size such that the average batch size was 24 (18 for LSTM-free models). We use Adam [24] with initial learning rate of 0.0003, decayed by 0.5 when validation WER did not improve over 10 epochs initially and 5 epochs after the first decay.

[3] To overcome trainability issues and encourage the optimizer to adjust the variance parameter, we found it necessary to re-parametrize it using $\tau^2 = \sigma$ and optimize τ via back-propagation.

Table 1: *Comparison to baselines. Training speed (char/sec) was measured on a GTX 1080 Ti GPU.*

model	dev WER	test WER	char/sec
pyramidal	15.83	16.16	1.1k
LSTM/NiN	14.57	14.70	1.1k
stacked hybrid	16.38	17.48	2.4k
interleaved hybrid	15.29	16.71	1.5k

6. Quantitative Results

6.1. Comparison to Baselines

The first set of experiments compares the proposed hybrid models to the baselines. The results are summarized in Table 1. We observe similar word error rates, with the interleaved model outperforming the stacked model and outperforming the pyramidal LSTM baseline on the development data but not the test data. The LSTM/NiN baseline was strongest. In terms of training speed, the stacked model is fastest by a large margin, followed by the interleaved model and the LSTM/NiN model. To confirm that the attention mechanism is actually contributing to the hybrid model and not just passing on activations, we performed a sanity check by training a stacked hybrid model with attention scores off the diagonal set to $-\infty$, and observed a drop of 1.25% absolute WER.

6.2. Position Modeling

Next, we evaluate the different approaches to position modeling (§ 4.2). The results are summarized in Table 2. When using additive positional encodings the model diverged, while concatenating embeddings converged, albeit to rather poor optima. The key/query positional embeddings in isolation diverged, and combination with concatenated input embeddings did not improve results. Only the hybrid models were able to obtain results comparable to the baselines. We also tried combining hybrid models with positional embeddings, but did not see improvements over the model without positional embeddings.

Table 2: *WER results on position modeling.*

model	dev	test
add (trig.)	diverged	
concat (trig.)	30.27	38.60
concat (emb.)	29.81	31.74
stacked hybrid	16.38	17.48
interleaved hybrid	15.29	16.71

Table 3: *WER results on attention biasing.*

model	dev	test
stacked hybrid	16.38	17.48
+ local masking	15.42	16.17
+ Gauss mask (init. small)	16.05	16.96
+ Gauss mask (init. large)	14.90	15.89
interleaved hybrid	15.29	16.71
+ local masking	15.44	16.19
+ Gauss mask (init. small)	16.43	16.89
+ Gauss mask (init. large)	15.00	15.82

6.3. Attention Biasing

This set of experiments tests the effect of introducing explicit attention biases that enable the model to control its context range (§ 4.3). The local diagonal mask was set to constrain the context to a window of 5 time steps, and Gaussian biasing variances were initialized to 9 (small setting) or 100 (large setting). Results are summarized in Table 3. For the stacked model, it can be seen that the biasing helps in general. The strongest model variant was the learnable Gaussian mask. Interestingly, it was important to initialize the Gaussian to possess a large variance. We hypothesize that this improves gradient flow early on in the model training, similar to how initializing LSTM forget gate biases to 1 (no forgetting) improves results [25]. The interleaved hybrid model shows similar trends. Note that the sometimes inconsistent ordering between dev and test results can be explained by the fact that the TEDLIUM dev set is relatively small with only 500 utterances.

The Gaussian mask allows inspecting its trainable variance parameter. Fig. 2 shows how the parameter evolves when initialized to a large value. It can be seen that in the first layer, diversity seems to be desirable, with some attention heads focusing on a small local context, and others on larger contexts. In contrast, the second layer does not appear to benefit from limiting its context. This partly confirms the idea of hierarchical modeling, where the modeling granularity increases across layers, but also shows that even at the bottom layer a controlled amount of long-range context is desirable.

7. Interpretability of Attention Heads

We hypothesize that certain attention heads respond to certain types of acoustic events. To test this hypothesis, we correlate the average attention that each attention head places on frames with the corresponding phoneme labels obtained via forced decoding. We re-train the stacked hybrid model with phonemes instead of characters as targets, and use encoder-decoder attention scores, summed over phoneme types, to obtain a soft alignment of phoneme labels for each frame. This gives us a measure for how much each frame in the sequence corresponds to the phoneme type under inspection.

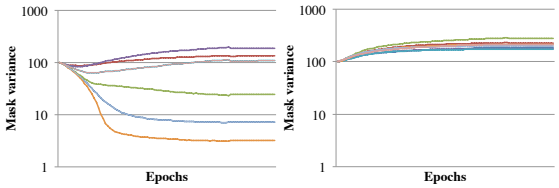


Figure 2: *Evolution of the variance parameters for each of the 8 attention heads over course of training (left: first layer, right: second layer).*

Table 4: *Analysis of function of attention heads. Note that we conducted a small amount of cherry picking by removing 4 outliers that did not seem to fit categories (OY from head 1, ZH from head 3, EH and ER from head 7). Entropy is computed over the correlation scores, truncated below 0.*

i	top phonemes	entropy	comments
1	S, TH, Z	3.7	sibilants
2	</s>	1.9	silence
3	UW, Y, IY, IX	3.6	”you” diphthong
	B, G, D		voiced plosives
	M, NG, N		nasals
4	XM, AW, AA, AY, L, AO, AH	3.2	A, schwa
5	ZH, AXR, R	3.5	R, ZH
6	ZH, Z, S	3.2	sibilants
	IY, IH, Y, UW		”you” diphthong
7	S, </s>, TH CH, SH, F	3.4	fricative, noise
8	mixed	3.7	unfocused

We now correlate these phoneme activations to each of the first layer's 8 attention heads. We average the matrices across rows to obtain the overall attention that each frame receives. We then compute the Pearson correlation coefficient of the summarized self-attention and encoder-decoder attention sequences, concatenated over utterances.

Table 4 shows the most highly correlated phonemes for each attention head, along with an attempt to classify these manually according to linguistic categories. This works remarkably well and we can clearly see a linguistically plausible division of labor, even though categories are neither exhaustive nor disjunct. Notice that head 2 seems to always focus on the utterance end where we usually expect silence, and head 8 is mostly unfocused, which we may interpret as these heads establishing channel and speaker context.

8. Conclusion

Applying self-attention to acoustic modeling is challenging for computational and modeling reasons. We investigate ways to address these challenges and obtain our best results when using a hybrid model architecture and Gaussian biases that allow controlling context range. This model is almost as good as a strong LSTM-based baseline at much faster computation speed. We highlight interpretability as an advantage over conventional models. Future work includes investigation of self-attention with other sequences of low-information states such as characters, and of transferring results on controlling context range and interpretability to text modeling.

9. References

[1] M. Ostendorf, "Moving beyond the beads-on-a-string'model of speech," in *Automatic Speech Recognition & Understanding (ASRU)*, 1999, pp. 79–84.

[2] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Annual Conference of the International Speech Communication Association (InterSpeech)*, 2015.

[3] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[4] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A Decomposable Attention Model for Natural Language Inference," in *Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA, 2016, pp. 2249–2255.

[5] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A Structured Self-attentive Sentence Embedding," in *International Conference on Representation Learning (ICLR)*, 2017, pp. 1–15.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Neural Information Processing Systems Conference (NIPS)*, 2017, pp. 5998–6008.

[7] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[8] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A Time-Restricted Self-Attention Layer for ASR," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[9] N. Kalchbrenner and P. Blunsom, "Recurrent Continuous Translation Models," in *Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, Washington, USA, 2013, pp. 1700–1709. [Online]. Available: http://www.aclweb.org/anthology/D13-1176

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, Montréal, Canada, 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *International Conference on Representation Learning (ICLR)*, San Diego, USA, 2015.

[12] Y. Zhang, W. Chan, and N. Jaitly, "Very Deep Convolutional Networks for End-to-End Speech Recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[13] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv:1607.06450*, 2016.

[14] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," *arXiv:1705.03122*, 2017.

[15] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding," in *Conference on Artificial Intelligence (AAAI)*, 2018.

[16] J. Im and S. Cho, "Distance-based Self-Attention Network for Natural Language Inference," *arXiv:1712.02047*, 2017.

[17] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks," in *International Conference on Language Resources and Evaluation (LREC)*, 2014, pp. 3935–3939.

[18] G. Neubig, M. Sperber, X. Wang, M. Felix, A. Matthews, S. Padmanabhan, Y. Qi, D. S. Sachan, P. Arthur, P. Godard, J. Hewitt, R. Riad, and L. Wang, "XNMT: The eXtensible Neural Machine Translation Toolkit," in *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston, 2018. [Online]. Available: https://arxiv.org/pdf/1803.00188.pdf

[19] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, and Others, "The Kaldi speech recognition toolkit," in *Workshop on Automatic Speech Recognition & Understanding (ASRU)*, 2011.

[20] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in *Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, 2015, pp. 1412–1421. [Online]. Available: http://aclweb.org/anthology/D15-1166

[21] Y. Gal and Z. Ghahramani, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," in *Neural Information Processing Systems Conference (NIPS)*, Barcelona, Spain, 2016, pp. 1019–1027. [Online]. Available: http://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-recurrent-neural-networ pdf

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Computer Vision and Pattern Recognition*, 2016. [Online]. Available: https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2016/papers/Szegedy{_}Rethinking{_}the{_}Inception{_}CVPR{_}2016{_}paper.pdf

[23] T. Q. Nguyen and D. Chiang, "Improving Lexical Choice in Neural Machine Translation," in *North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA, 2018. [Online]. Available: https://arxiv.org/pdf/1710.01329.pdf

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf

[25] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An Empirical Exploration of Recurrent Network Architectures," in *International Conference on Machine Learning (ICML)*, Lille, France, 2015.