Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions

Awni Hannun, Ann Lee, Qiantong Xu, Ronan Collobert

Facebook AI Research

awni@fb.com, annl@fb.com, qiantong@fb.com, locronan@fb.com

Abstract

We propose a fully convolutional sequence-to-sequence encoder architecture with a simple and efficient decoder. Our model improves WER on LibriSpeech while being an order of magnitude more efficient than a strong RNN baseline. Key to our approach is a time-depth separable convolution block which dramatically reduces the number of parameters in the model while keeping the receptive field large. We also give a stable and efficient beam search inference procedure which allows us to effectively integrate a language model. Coupled with a convolutional language model, our time-depth separable convolution architecture improves by more than 22% relative WER over the best previously reported sequence-to-sequence results on the noisy LibriSpeech test set.

Index Terms: speech recognition, sequence-to-sequence, neural networks

1. Introduction

Sequence-to-sequence models with attention have been used for speech recognition [1] since their inception in machine translation [2, 3, 4]. These models have yielded state-of-the-art results in some settings [5], however; approaches such as CRF style end-to-end models [6, 7] and more traditional HMM based models [8] are often superior.

While sequence-to-sequence models sometimes generalize well in speech recognition, they often come with a big hit to efficiency. The encoder typically consists of several layers of large bidirectional LSTMs [9, 10]. The decoder also uses a number of inefficient and sequential techniques. Efficiency is useful for fast training and evaluation times and is critical to the massive scale used in the semi-supervised and weakly supervised regimes [11, 12].

In this work we develop a highly efficient sequence-to-sequence model which gives state-of-the-art results for non speaker adapted models on both LibriSpeech test sets [13]. Key to our approach is a *fully convolutional* encoder with a time-depth separable (TDS) block structure. Our TDS convolution improves in WER over an RNN baseline and due to the parallel nature of the computation is much more efficient. We also discard slow and sequential techniques previously thought to be important to the accuracy of these models. These include neural content attention, location based attention, and scheduled sampling. In turn, we give more efficient alternatives.

Also key to our approach is a highly efficient and stable beam search inference procedure. Unlike previous work [14], accuracy does not degrade with very large beam sizes. This enables us to better leverage the constraint of a convolutional language model which gives substantial improvements in WER over a simple n-gram baseline.

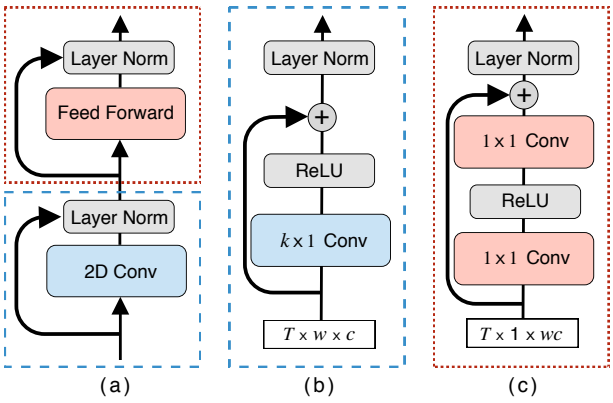

Figure 1: The TDS convolution model architecture. (a) The sub-blocks of the TDS convolution layer are (b) a 2D convolution over time followed by (c) a fully connected block.

2. Model

We consider an input utterance $X = [X_1, \ldots, X_T]$ and an output transcription $Y = [y_1, \ldots, y_U]$. The sequence-to-sequence model *encodes* X into a hidden representation and then *decodes* the hidden representation into a sequence of predictions for each output token. The encoder is given by

$$\begin{bmatrix} K \\ V \end{bmatrix} = \text{encode}(X) \tag{1}$$

where $K = [K_1, \ldots, K_T]$ are the keys and $V = [V_1 \ldots, V_T]$ are the values. The decoder is given by

$$Q_u = g(y_{u-1}, Q_{u-1}) \tag{2}$$

$$S_u = \text{attend}(Q_u, K, V) \tag{3}$$

$$P(y_u \mid X, y_{<u}) = h(S_u, Q_u) \tag{4}$$

Here $g(\cdot)$ is an RNN which encodes the previous token and query vector Q_{u-1} to produce the next query vector. The attention mechanism attend$(\cdot)$ produces a summary vector S_u, and $h(\cdot)$ computes a distribution over the output tokens.

2.1. Time-Depth Separable Convolutions

Our proposed time-depth separable (TDS) convolution block (see Figure 1) partially decouples the aggregation over time from the mixing over channels. This allows us to increase the receptive field of the model with a negligible increase in the number of parameters. In preliminary experiments we find that the TDS convolution block generalizes much better than other deep convolutional architectures [6, 15] and needs fewer parameters. Another benefit of our block structure is it can be implemented efficiently using a standard 2D convolution.

The block starts with a layer of 2D convolution which operates over an input of shape $T \times w \times c$ and produces an output of shape $T \times w \times c$ where T is the number of time-steps, w is the input width and c is the number of input (and output) channels. The kernels are size $k \times 1$. The total number of parameters in this layer is kc^2 which can be made small by keeping c small. We follow the convolution with a ReLU non-linearity.

We then view the output of the convolution as $T \times 1 \times wc$ and apply a fully-connected layer, which is a sequence of two 1×1 convolutions (i.e. linear layers) with a ReLU non-linearity in between. We add residual connections [10, 16] and layer normalization [17] after the convolution and the fully connected layer. The layer normalization is over all dimensions for a given example including time.

The TDS architecture has three sub-sampling layers each with a stride of 2 for a total sub-sampling factor of 8. We also increase the the number of output channels at each sub-sampling layer since we compress the information in time. For simplicity these layers do not have residual connections and are only followed by a ReLU and layer normalization.

2.2. Efficient Decoder

The decoder is sequential in nature since to compute the next output requires the previous prediction. However, at training time we use teacher forcing–the previous ground truth is used in place of the previous prediction. In principle, this allows us to compute all output frames simultaneously. The outputs of the RNN given by $g(\cdot)$ cannot be computed in parallel, however; unrolling the computation and making a single call to an efficient CuDNN [18] implementation is much faster than calling U separate kernels. After the following optimizations, the decoder accounts for less than 10% of the total iteration time.

Techniques such as scheduled sampling [19], input feeding [20] and location-based attention [1] introduce a sequential dependency in the decoder. We discard these techniques in favor of approaches which can be computed in parallel. We simply do not use input feeding and location-based attention as we find that we can achieve good WERs without them. We replace scheduled sampling with random sampling (section 2.2.1).

We use an inner-product key-value attention which can be implemented much more efficiently than a neural attention. For a single example the attention is given by

$$S = V \cdot \text{softmax}\left(\frac{1}{\sqrt{d}} K^\top Q\right) \tag{5}$$

We scale the inner products by the inverse square root of their hidden dimension d. This improves convergence and helps the model learn an alignment. However, we do not see a consistent improvement in generalization [21].

2.2.1. Random Sampling

Scheduled sampling [19] limits exposure bias by bringing the training conditions closer to the testing conditions. However, it introduces a sequential dependency in the decoder, since it sometimes uses the previous prediction at the next time-step.

Instead, we propose random sampling, where the previous prediction is replaced with a randomly sampled token [22]. First we decide with probability P_{rs} to sample a given input token. If we sample, then choose a new token from a uniform distribution. This allows us to vectorize the implementation as follows:

1. Sample U random numbers c_j uniformly from $[0, 1]$.
2. Set $R = [r_1, \ldots, r_U]$ where $r_j = \mathbb{I}(c_j > P_{\text{rs}})$ and P_{rs} is the sampling probability.
3. Sample a vector Z of U tokens. We use a uniform distribution over the output tokens not including end-of-sentence (EOS).
4. Construct $\hat{Y} = R \circ Z + (1 - R) \circ Y$.

As we show later, random sampling improves WER.

2.3. Soft Window Pre-training

We propose a simple soft attention window pre-training scheme to enable the training of very deep convolutional encoders. Compared to prior work [23], our approach is simple to implement, results in negligible additional computational expense, and needs very little tuning.

We encourage the model to align the output at uniform intervals along the input by penalizing attention values which are too far from the desired locations. Let W be a $T \times U$ matrix with entries $W_{ij} = (i - \frac{T}{U} j)^2$. The matrix W encodes the (squared) distance between the i-th input and the j-th output assuming the outputs are spaced at uniform intervals along the input – hence the scaling factor T/U. We apply W to the attention as follows

$$S = V \cdot \text{softmax}\left(\frac{1}{\sqrt{d}} K^\top Q - \frac{1}{2\sigma^2} W\right) \tag{6}$$

The term σ is a hyper-parameter which dampens the effect of W. The application of W is equivalent to multiplying the normalized attention vector (i.e. after the softmax) by a Gaussian shaped mask. In that respect, σ is simply the standard deviation of the Gaussian.

We use the window pre-training for the first few epochs and then switch it off. This is sufficient to enable the model to learn an alignment and converge. In general σ does not need to be tuned when model hyper-parameters change. An exception is when the amount of sub-sampling in the encoder changes, σ should change accordingly.

2.4. Regularization

We use three additional forms of regularization to control overfitting and improve the generalization of the model.

2.4.1. Dropout

First we apply dropout [24] after each layer in each block of the encoder. We apply dropout after the non-linearity and prior to layer normalization. We do not use any dropout in the decoder.

2.4.2. Label Smoothing

We use label smoothing [25] to reduce over-confidence in predictions. As in machine translation [21], we find that label smoothing hurts loss on the dev set but improves WER.

2.4.3. Word Piece Sampling

We use word pieces [26] as outputs following the Unigram Language Model approach [27]. During training, we sample word piece representations for a given transcription [27], but unlike prior work, we sample at the word-level instead of the sentence-level. For each word, with probability $1 - P_{\text{wp}}$ we take the most likely word piece representation or with probability P_{wp} uniformly sample over the top-ten most likely alternatives.

3. Beam Search Decoding

We use an *open-vocabulary* beam search decoder which optimizes the following objective

$$\log P_{\text{s2s}}(Y \mid X) + \alpha \log P_{\text{LM}}(Y) + \beta |Y| \tag{7}$$

The term $|Y|$ counts the number of tokens in Y. In the above, α is the LM weight and β is the token insertion term.

3.1. Stabilizing Beam Search

Sequence-to-sequence beam search decoders are known to be unstable sometimes exhibiting worse performance with an increasing beam size [14]. We use two techniques to stabilize the beam search. This allows our model to extract more value from the integration of an LM, since we can use a large beam size to effectively search over the space of possible hypotheses.

3.1.1. Hard Attention Limit

We do not allow the beam search to propose any hypotheses which attend more than t_{max} frames away from the previous attention peak. In practice we find that t_{max} only needs to be tuned once for a given data set and can otherwise remain unchanged.

3.1.2. End-of-sentence Threshold

In order to bias the search away from short transcriptions, we only consider end-of-sentence (EOS) proposals when the score is greater than a specified factor of the best candidate score

$$\log P_u(\text{EOS} \mid y_{<u}) > \gamma \cdot \max_c \log P_u(c \mid y_{<u}) \qquad (8)$$

Like the hard attention limit, we find the parameter γ only needs to be tuned once for a given data set.

3.2. Efficiency

We use a few heuristics to further improve the efficiency of the beam search. First, we set a beam threshold [6] to prune hypotheses in the beam which are below a fixed range from the best hypothesis so far.

We also apply a threshold when proposing new candidate tokens to the current set of hypotheses in the beam. Similar to Equation 8, we require that the proposed token score satisfy

$$\log P_u(y \mid y_{<u}) > \max_c \log P_u(c \mid y_{<u}) - \eta \qquad (9)$$

Finally, we batch compute the updated set of probabilities for every candidate in the beam, so only one forward pass is required at each step. These techniques result in a fast decoding time even with a deep convolutional LM and a large beam.

4. Experiments

We perform experiments on the full 960-hour LibriSpeech corpus [13]. Our best encoder has two 10-channel, three 14-channel and six 18-channel TDS blocks. We use three 1D convolutions to sub-sample over time, one as the first layer and one in between each group of TDS blocks. Kernel sizes are all 21×1. A final linear layer produces the 1024-dimensional encoder output. The decoder is a one-layer GRU with 512 hidden units. Weights are initialized from a uniform distribution $\mathcal{U}(-\sqrt{4/f_{in}}, \sqrt{4/f_{in}})$, where f_{in} is the fan-in to each unit.

Input features are 80-dimensional mel-scale filter banks computed every 10-ms with a 25-ms window. We use 10k word pieces computed from the *SentencePiece* toolkit [28] as the output token set. All models are trained on 8 V100 GPUs with a batch size of 16 per GPU. We use synchronous SGD with a learning rate of 0.05, decayed by a factor of 0.5 every 40 epochs. We clip the gradient norm to 15. The model is pre-trained for three epochs with the soft window and $\sigma = 4$. We use 20% dropout, 5% label smoothing, 1% random sampling and 1% word piece sampling.

Table 1: A comparison of the TDS conv model to other models on the Librispeech Dev and Test sets.

Model	Dev WER		Test WER	
	clean	other	clean	other
hybrid, speaker adapted				
CAPIO (single) [33] + RNN	3.12	8.28	3.51	8.58
CAPIO (ensemble) [33] + RNN	2.68	7.56	3.19	7.64
CNN ASG [31] + ConvLM	3.16	10.05	3.44	11.24
RNN S2S [23]	4.87	14.37	4.87	15.39
RNN S2S [23] + 4-gram	4.79	14.31	4.82	15.30
RNN S2S [23] + LSTM	3.54	11.52	3.82	12.76
TDS conv	5.04	14.45	5.36	15.64
TDS conv + 4-gram	3.75	10.70	4.21	11.87
TDS conv + ConvLM	3.01	8.86	3.28	9.84

We train two word piece LMs on the 800M-word text-only data set. The first is a 4-gram trained with KenLM [29] and the second is a convolutional LM (ConvLM) [30] using the same model architecture and training strategy as [31]. We use a beam size of 80, set $t_{\max} = 30$, the EOS penalty $\gamma = 1.5$ and $\eta = 10$. The LM weight and token insertion terms are cross-validated with each dev set and LM combination. We use the *wav2letter++* framework to train and evaluate our models [32].

4.1. Results

Table 1 compares the TDS model with three other systems. The CAPIO system is a hybrid HMM-DNN with speaker adaptation [33]. The other two are end-to-end models, one using the CRF-style ASG loss [31] and the other a sequence-to-sequence model with an RNN encoder [23].

Our proposed model achieves a state-of-the-art for end-to-end systems of 3.28 WER on test clean and 9.84 WER on test other. Compared with the RNN-based encoder [23], the TDS model improves WER by 14.1% on test clean and 22.9% on test other with nearly a factor of 4 reduction in parameters (136M vs. 37M). The TDS model benefits more from an external LM. This could be due to (1) a better loss on the correct transcription and (2) a more effective beam search.

4.2. Model Variations

Table 2 shows results from varying the number of TDS blocks, the number of parameters, the word piece sampling probability and the amount of random sampling. For each setting we train three models and report the best and the average WER.

We reduce the number of parameters without changing the receptive field by reducing the number of channels in each group of TDS blocks from (10, 14, 18) to (10, 12, 14) or (10, 10, 10). The model is very sensitive to decreasing the number of parameters. We also examine the effect of varying the number of TDS blocks without changing the number of parameters or the receptive field. For 9 TDS blocks we use (14, 16, 20) channels with $k = 27$, and for 12 TDS blocks we use (10, 16, 16) channels with $k = 19$. We show that a small amount of word piece sampling is helpful. With a higher P_{wp} the model sometimes converges poorly, likely due to the variability in the targets. A small amount of random sampling is also helpful. Finally, when we remove soft window pre-training, the model takes much longer to converge and achieves a worse result. The soft window clearly helps guide the attention early in training.

Figure 2 shows the effect of the receptive field on WER.

Table 2: The sensitivity of our model to architecture and regularization hyper-parameters. The parameter N is the number of TDS blocks, P_{wp} is the word piece sampling rate, and P_{rs} is the random sampling rate. Missing entries correspond to the value in the first row. We report the lowest WER over three runs along with the mean in parentheses using a beam size of 1 and no LM.

N	params ($\times 10^6$)	P_{wp}	P_{rs}	Dev Clean	Dev Other
11	36.5	1%	1%	5.04 (5.13)	14.45 (14.77)
	24.4			5.36 (5.45)	15.16 (15.24)
	14.9			5.95 (5.99)	16.25 (16.44)
9				5.18 (5.27)	15.34 (15.37)
12				5.10 (5.33)	14.99 (15.26)
		0%		5.25 (5.32)	14.89 (15.00)
		2%		5.04 (5.46)	14.88 (15.41)
			0%	5.08 (5.24)	15.00 (15.21)
			5%	5.11 (5.25)	14.65 (14.80)
No soft window pre-training				5.55 (5.58)	14.99 (15.30)

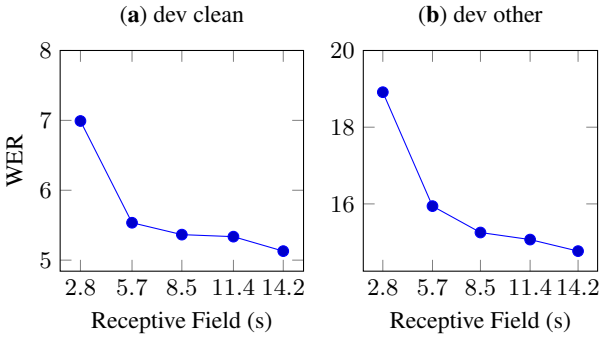

Figure 2: The WER as a function of the receptive field. We vary the kernel size, $k \in \{5, 9, 13, 17, 21\}$, otherwise every model has ∼36.5 million parameters. We report the mean WER over three runs using a beam size of 1 and no LM.

There is a sharp increase in WER when the size of the receptive field drops below a threshold. Qualitative analysis shows that the high WER is often due to catastrophic errors such as looping and skipping, a common problem for sequence-to-sequence models [14]. We hypothesize that without a large receptive field, the encoder keys do not have enough context to disambiguate queries from the decoder.

Figure 3 shows how WER changes with the size of the beam. While most of the gain from including an external LM comes even at small beam size, we see consistent improvements up to a beam size of 80, particularly on dev other.

4.3. Efficiency

We compare the TDS conv model to a strong RNN baseline in terms of training efficiency on LibriSpeech [23]. The RNN baseline encoder consists of six bidirectional LSTMs. Both models have a total sub-sampling factor of 8. Our best TDS architecture can complete one full epoch over the LibriSpeech training set in 7 minutes. This is more than 10× faster than our implementation of the RNN baseline and more than 4× faster than the RNN baseline encoder but with the efficient decoder described in Section 2.2.

Our beam search runs at an average rate of 0.57 and 0.93 seconds-per-sample on dev clean and other with the 4-gram LM and a beam size of 80. With the ConvLM, times increase to 0.73 and 1.20 seconds-per-sample at the same beam size.

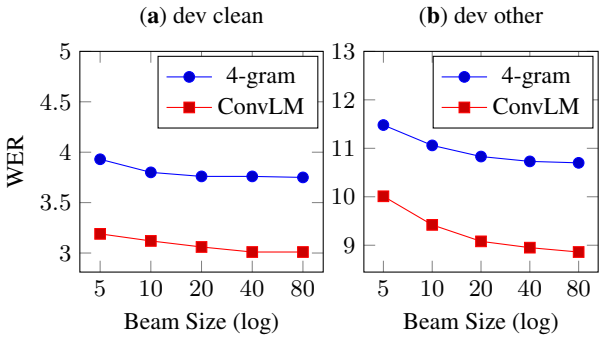

Figure 3: The WER as a function of beam size for both the 4-gram and the convLM.

5. Related Work

Our work builds on a large body of work aimed at improving sequence-to-sequence models with attention for both speech recognition [5, 14, 23] and other application domains. Fully convolutional encoders have worked well in machine translation [15]. They have also given state-of-the-art results in speech recognition [31] with more structured loss functions like the AutoSegCriterion [6]. However, we are not aware of any competitive results with fully convolutional encoders in sequence-to-sequence models for speech recognition.

The high-level encoder architecture is similar to the Transformer model [21]; however, we consider convolutions instead of self-attention. Our architecture is inspired by and quite related to the lightweight convolution [34]. An important idea of that work and ours is the separation of the integration over time from the mixing over channels which improves both accuracy and efficiency. Other than the application to speech, some differences in our encoder architecture are (1) the time-depth separable convolution can be implemented with a simple 2D convolution and (2) our models do not use any normalization over the time dimension of the kernels.

Depth-wise separable convolutions have been used to improve the efficiency and accuracy of computer vision models [35, 36]. The first layer of the TDS block can be seen as a grouped 1D convolution with cw channels, a group size of c, and weights tied between groups. Grouped convolutions have also been used in computer vision to improve efficiency for e.g. model-parallel training [37] and classification accuracy [38].

6. Conclusion

We have shown that a fully convolutional encoder and a simple decoder can give superior results to a strong RNN baseline while being an order of magnitude more efficient. Key to the success of the convolutional encoder is a time-depth separable block structure which allows the model to retain a large receptive field. We also show how to integrate a strong convolutional LM with a stable and scalable beam search procedure.

7. Acknowledgements

Thanks to Michael Auli, Abdelrahman Mohamed, Tatiana Likhomanenko and Gabriel Synnaeve for helpful conversations.

8. References

[1] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[2] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, 2015.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[5] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[6] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[7] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[8] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," IEEE Signal Processing Society, Tech. Rep., 2011.

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

[10] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4845–4849.

[11] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," *arXiv preprint arXiv:1808.09381*, 2018.

[12] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 181–196.

[13] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[14] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *arXiv preprint arXiv:1612.02695*, 2016.

[15] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[18] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[19] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[20] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[22] X. Wang, H. Pham, Z. Dai, and G. Neubig, "SwitchOut: an efficient data augmentation algorithm for neural machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018, pp. 856–861.

[23] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *Interspeech*, 2018.

[24] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[26] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[27] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[28] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[29] K. Heafield, "KenLM: Faster and smaller language model queries," in *Proceedings of the sixth workshop on statistical machine translation*. Association for Computational Linguistics, 2011, pp. 187–197.

[30] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 933–941.

[31] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully convolutional speech recognition," *arXiv preprint arXiv:1812.06864*, 2018.

[32] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert, "wav2letter++: The fastest open-source speech recognition system," *arXiv preprint arXiv:1812.07625*, 2018.

[33] K. J. Han, A. Chandrashekaran, J. Kim, and I. Lane, "The CAPIO 2017 conversational speech recognition system," *arXiv preprint arXiv:1801.00059*, 2017.

[34] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," *arXiv preprint arXiv:1901.10430*, 2019.

[35] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[38] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.