INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Scaling Up Online Speech Recognition Using ConvNets

Vineel Pratap[1], *Qiantong Xu*[1], *Jacob Kahn*[1], *Gilad Avidov*[1], *Tatiana Likhomanenko*[1], *Awni Hannun*[2], *Vitaliy Liptchinsky*[1], *Gabriel Synnaeve*[2], *Ronan Collobert*[1]

[1]Facebook AI Research, Menlo Park
[2]Facebook AI Research, NYC

{vineelkpratap,qiantong,jacobkahn,avidov,antares,awni,vitaliy888,gab,locronan}@fb.com

Abstract

We design an online end-to-end speech recognition system based on Time-Depth Separable (TDS) convolutions and Connectionist Temporal Classification (CTC). We improve the core TDS architecture in order to limit the future context and hence reduce latency while maintaining accuracy. The system has almost three times the throughput of a well tuned hybrid ASR baseline while also having lower latency and a better word error rate. Also important to the efficiency of the recognizer is our highly optimized beam search decoder. To show the impact of our design choices, we analyze throughput, latency, accuracy, and discuss how these metrics can be tuned based on the user requirements.

Index Terms: online speech recognition, low latency

1. Introduction

The process of transcribing speech in real-time from an input audio stream is known as online speech recognition. Most automatic speech recognition (ASR) research focuses on improving accuracy without the constraint of performing recognition in real time. For certain applications such as live video captioning or on-device transcription, however, low latency speech recognition is essential. In these cases, online speech recognition with a limited time delay is needed to provide a good user experience.

Furthermore, deployment of a real-time speech recognition system at scale poses a number of challenges. First, the system must be computationally efficient to minimize the required hardware resources and to handle a large number of concurrent requests. Second, the system needs to minimize the time between a spoken word appearing in the audio and the corresponding text produced by the system. Third, the deployed recognizer should be competitive in accuracy with an offline research-grade system. We measure the three aforementioned constraints using the following metrics: (i) throughput, (ii) Real-Time Factor (RTF) and latency, and (iii) Word Error Rate (WER).

We discuss the design of a novel ASR system that achieves 3x better throughput compared to a strong hybrid baseline. We use Time-Depth Separable convolutions [1] as the building block for our acoustic model and an efficient beam-search decoder provided by the wav2letter++ open-source ASR toolkit [2].

The rest of the paper is organized as follows. Section 2 presents related work on low latency online ASR. In Section 3, we review the design principles of our system, including acoustic models, the decoder and present novel ideas to reduce latency and greatly improve computational efficiency. In Section 4 we discuss our experimental setup: benchmarks for measuring throughput and accuracy, as well as hardware configuration. In Section 5 we present out experimental results and ablative analysis. Finally, in Section 6 we discuss future work and conclude.

2. Related Work

Taking an ASR system from research to a low-latency, high-throughput deployment while maintaining low WER involves non-trivial changes to the implementation. For example, many research systems use bi-directional RNNs [3, 4] or Transformers [5] which uses (self-)attention [6, 7, 8, 9] which require unlimited future context and make low-latency deployment impossible. Fully convolutional acoustic models can be very efficient to train, but are often used with large future context sizes [10]. Here, we build on a large body of work in making research-grade ASR systems work in resource constrained low-latency environments.

Our architecture uses Time-Depth Separable convolution (TDS) [1] as the core building block. Bi-directional RNNs and Transformers either require considerable changes for low-latency deployment [11, 12] or degrade rapidly when limiting the amount of future context [8]. While latency controlled bi-directional LSTMs are commonly used in online speech recognition [13], incorporating future context with convolutions can yield more accurate and lower latency models [14]. We find that the TDS convolution can maintain low WERs with a limited amount of future context.

Some recent work in low-latency speech recognition uses the RNN Transducer (RNN-T) [15] as the core architecture [16, 17]. Unlike attention models, the RNN-T decoder is causal and can be easily streamed since it does not rely on future context. Departing from prior work, we find that a CTC trained model can yield better WER than an RNN-T model while being simpler and more efficient to deploy.

Depth-wise separable convolutions [18] have been used in domains such as computer vision [19] and machine translation [20] yielding dramatic reductions in model size and computational throughput while maintaining accuracy. The TDS architecture we use here also results in much lighter weight yet still low WER models. Other architectures, such as the Time-Channel Separable convolution have also shown similar gains in computational efficiency with little if any hit to accuracy [21].

3. Technical Details

In this section we describe the architectural design, algorithmic choices and various performance optimizations of our speech recognizer[1].

[1]Training recipe along with inference code is available under https://github.com/facebookresearch/wav2letter

http://dx.doi.org/10.21437/Interspeech.2020-2840

3.1. Low-latency acoustic models

Our acoustic models are based on Time-Depth Separable (TDS) convolutions [1]. The reasons for choosing these models is two-fold. First, TDS blocks use grouped convolutions which dramatically reduce the number of parameters while still achieving low WER. This makes inference computationally efficient and keeps the model size small. Second, limiting the future context for convolution operations in TDS, which is necessary for maintaining low latency, leads to a small WER degradation.

Figure 1(a) shows the architecture of the TDS Block used in our work. The TDS block we use is modified from the original TDS implementation to make the architecture streaming friendly. The changes are described below.

Asymmetrically Padded Convolutions: The latency of 1-D convolutions is impacted by the number of future inputs needed to generate the current output. In order to minimize this, we use asymmetric padding for convolutions which adds more (zero) padding at the start of the input. This reduces the dependency of the current output on the future input and hence reduces the latency of the model.

For example, consider a TDS(10, 9, 80, 4) block (See Fig. 1(a) for notation) which uses symmetric padding. In order to generate the first output frame, the 1-D convolution with a filter size of 9 needs 5 input frames, since 4 frames will be padded to the start of the input. We can reduce this future context dependency to just 2 frames if we use a TDS(10, 9, 80, 1) block which pads with 7 frames at the start of the input.

Removing the Time-Dependency of LayerNorm: The TDS Block in [1] performs layer normalization across all axes including time for a given sample. This makes the output depend on the full sample which makes streaming impractical. We resolve this by performing standard layer normalization [22] which normalizes across the width (w) and channels (c) axes. For an input x of shape $T \times w * c$, we compute the layernorm output $\hat{x}$ as

$$\hat{x}[i,j] = g * \frac{x[i,j] - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + b \quad i \in [0,T), j \in [0, w*c) \quad (1)$$

where $\mu_i = \frac{1}{w*c}\sum_j x[i,j]$, $\sigma_i^2 = \frac{1}{w*c}\sum_j (x[i,j] - \mu_i)^2$, g and b are scalar affine transform parameters and ϵ is a small constant added for numerical stability.

Figure 1(b) shows the architecture of our acoustic model. The model consists of two 15-channel, three 19-channel, four 23-channel and five 27-channel TDS blocks. They are separated by 1-D convolution layers which increase the number of output channels and optionally perform subsampling. A final linear layer produces an N-dimensional output (N is the size of the token set) which is later passed to a log-softmax layer prior to computing the CTC loss. The model has a total of 104 million parameters and has a total subsampling factor of 8. It takes 80-dimensional log mel-scale filter bank features as input. The features are extracted with a stride of 10ms so the model has a receptive field of ∼10 seconds per frame and a future context of 250ms.

3.2. Online Beam Search Decoding

To integrate a language model, we develop an online version of the beam search decoder based on wav2letter++ [2]. The online decoder consumes the encoded frames for the current input audio chunk and extends the beam search graph. We output the most likely sequence of words based on this extended beam search graph for every chunk. Since the best path can change as we extend the beam search graph, we allow for correction of partial transcriptions generated earlier. Also, in order not to have the history in memory grow infinitely with incoming audio chunks, pruning is applied to the history buffer after consuming a certain number of chunks.

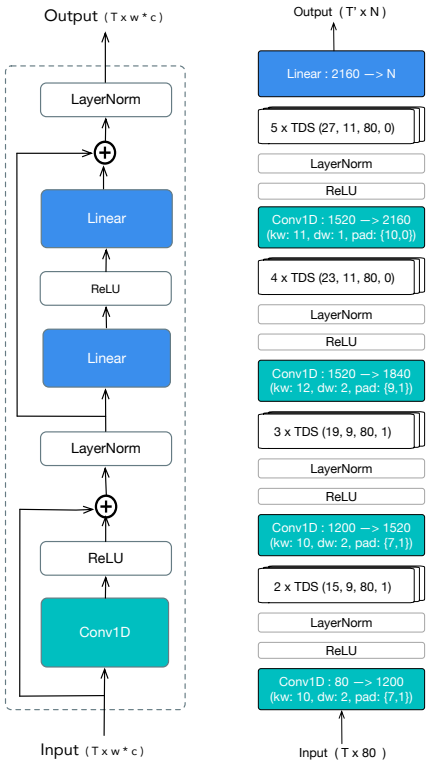

Figure 1: *(a) Time-Depth Separable Convolutional Block, TDS(c, kw, w, rPad). Params of Conv1D: in,out channels=w · c, stride=1, filter_size=kw, num_groups=w, padding={kw − 1−rPad, rPad}. Parameters of Linear: in,out channels=w∗c (b) AM Architecture. Notation: dw = stride, kw = filter size.*

Apart from the existing performance optimizations included in wav2letter++ decoder [10], we introduce two further pruning techniques. First, we consider only top-K (e.g. $K = 50$) tokens according to the acoustic model score, when expanding each hypothesis in the beam. This is also commonly known as acoustic pruning. Second, we propose only the blank symbol if its posterior probability is larger than 0.95 [23]. With these optimizations as well as the 8x reduction in the number frames from subsampling in the acoustic model, we find that decoding time accounts for about 5% of the overall inference procedure.

3.3. Inference Implementation

We wrote a standalone, modular platform to perform the full online inference procedure. The pipeline takes audio chunks as input and processes them in an online manner. To perform the matrix multiplications and 1-D group convolutions required by the acoustic model, we use the 16-bit floating point FBGEMM[2]

[2]https://github.com/pytorch/FBGEMM

implementation. We have carefully optimized memory use by relying on efficient I/O buffers and extensively profiled to remove any excess computational overhead.

4. Experimental Setup

4.1. Data

The training set used for our experiments consists of around 1 million utterances ($\sim$13.7K hours) of in-house English videos publicly shared by users. The data is completely anonymized and doesn't contain any personally-identifiable information (PII). We report results on two test sets - vid-clean and vid-noisy consisting of around 1.4K utterances each ($\sim$20 hours). More information about the dataset can be found in [24].

4.2. Training

All our experiments are run using wav2letter++ framework [2] with 64 GPUs for each experiment. We use 80-dimensional log mel-scale filter banks as input features, with STFTs computed on 25ms Hamming windows strided by 10ms. For the output token set, we use a vocabulary of 5000 sub-word tokens generated from SentencePiece toolkit [25]. We use local mean and variance normalization instead of global normalization on the input features prior to the acoustic model so that the system can run in an online manner. Local normalization computes the summary statistics over the prior n frames and uses them to normalize the input features at the current frame. We use $n = 300$ for all of our experiments which corresponds to a window size of approximately 3 seconds. We also use SpecAugment [3] for data augmentation in all of the experiments.

4.3. Baseline Systems

We compare our work with two strong baselines. All the systems (including ours) use the same train, valid and test sets.

Baseline 1 The first baseline [24] is a hybrid system based on context-dependent graphemes (chenones) and uses multi-layer Latency Controlled Bidirectional Long Short-Term Memory layers (LC-BLSTM)[26] in the acoustic model. The system is initially bootstrapped with cross entropy (CE) training and then fine-tuned with the lattice-free maximum mutual information (LF-MMI) [27] loss function.

Baseline 2 The second system [16] is based on the RNN-T [15] architecture and uses Latency Controlled Bidirectional Long Short-Term Memory layers (LC-BLSTM) [26] in the acoustic model. The token set consists of 200 sentence pieces, constructed with the sentence piece library [25]. Unlike Baseline 1, the second baseline is trained in an end-to-end manner.

4.4. Evaluation Benchmarks

For consistency in results, all the benchmarks, including baselines, are run on Intel Skylake CPUs with 18 physical cores and 64GB of RAM. We describe the performance metrics evaluated in our experiments below.

Real-Time Factor Real-Time Factor (RTF) is the ratio between the time taken to process the input and the input duration. For a system to be considered real-time, RTF should be $\leq$ 1. RTF is dependent on the number of concurrent streams being run by the system. Further, we define "RTF@40" as the RTF using 40 concurrent streams.

Throughput Throughput is defined as the rate at which audio is consumed. In other words, it is the number of audio seconds processed per wall clock second.

User-Perceived Latency User-perceived latency is the latency metric most relevant to the end user. It is affected by chunk size, RTF, acoustic model context, and decoder output delay. Quantifying latency as a function of these parameters is difficult. Instead we compute this latency in an empirical and end-to-end manner by measuring the timestamp of when a transcribed word is available to the user and compare this with the same word's timestamp in the original audio. More formally, we define user-perceived latency as the average delay between the end timestamp of the words in the reference transcript and time when it is shown to the user by the system.

As an example, consider a 1 second audio file with transcript "how are you" and the corresponding start and end word timestamps for the words as (100ms-200ms), (300ms-400ms), (500ms-600ms) respectively. Lets consider an ASR system with RTF 0.2 which processes the audio file and outputs the words "how", "are" after consuming 500ms of audio and the word "you" after looking at 1000ms of the audio. For this system, word "how" will be produced at 600ms (500ms for audio chunk to be available and 500ms $*$ RTF = 100ms for processing it) while the true timestamp of the end of the word is 200ms which gives us a latency of 400ms. Computing this for every word and taking the average, we can see that this system has average user-perceived latency of (400ms + 200ms + 500ms) / 3 = 366.67 ms.

We have measured user-perceived latency on a set of 1000 samples from the TIMIT [28] corpus, which comes with word alignments in advance. We note that all the ASR systems that we consider produce the correct transcript on these samples (WER = 0), which is necessary for this latency analysis.

5. Results and Analysis

Table 1 shows the performance comparison of our system with the two baselines mentioned in 4.3. We use 750ms chunk size for all the experiments with our system. User-perceived latency is measured at 40 concurrent streams. We can see that our system is able to achieve better WER at a much higher throughput even when using 16-bit floating-point precision as compared to the 8-bit fixed point precision used by the baselines.

Table 1: *A comparison of our system with the baseline systems.*

	Baseline 1 LC-BLSTM + LF-MMI	**Baseline 2** LC-BLSTM + RNN-T	**Our System** TDS + CTC
vid-clean WER	14.1	13.93	13.19
vid-noisy WER	22.15	22.58	21.16
Total Parameters	80 mil	60 mil	104 mil
Inference Precision	INT8	INT8	FP16
Throughput	55	64	147
RTF@40	0.70	0.60	0.26
User-perceived latency	1.18 sec	-	1.09 sec

5.1. Effect of low-latency acoustic models

We performed an ablation study to observe the change in WER from the original TDS model [1] with global input normalization to the low latency version we propose here. Table 2 shows that there is only a $\sim$4% relative increase in WER because of the changes we make to decrease the latency of the model. We observe that limiting the future context of the acoustic model contributes the most to the degradation in WER.

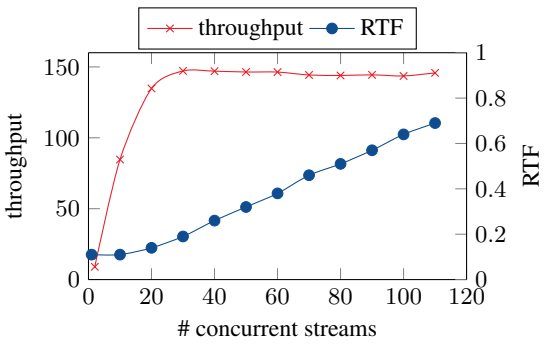

Figure 2: *RTF, throughput trade-off vs concurrent streams.*

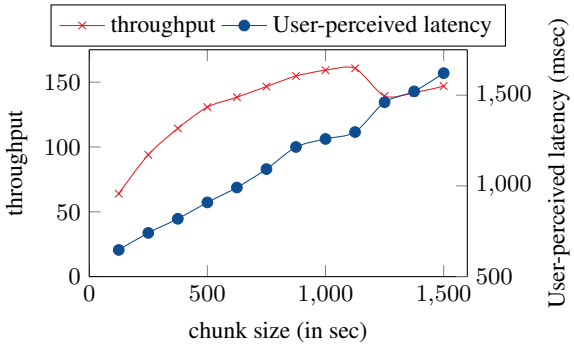

Figure 3: *Latency, throughput trade-off vs chunk size.*

Table 3 shows the effect of WER with varying future context sizes. We have kept the receptive field of our convolutional acoustic model fixed at 10 seconds, use local normalization and remove time from layer normalization axes for all the experiments. We see that the WER almost always improves as the future context size is increases.

However, with increasing future context size, the delay between audio and the corresponding transcript also increases. This is illustrated in figure 4 where the word timestamps generated from greedy decoding on an input audio file with only one spoken word is shown for varying right context sizes.

These results demonstrate that asymmetric padding is important for low latency convolutional acoustic models. We see a significant reduction in latency with just $\sim 4\%$ relative drop in WER going from a model with symmetric convolutions (5 seconds future context) to a model with asymmetric convolutions (250ms future context).

Table 2: *Effect on WER when decreasing TDS model latency.*

	vid-clean	**vid-noisy**
Original TDS with input globalnorm	12.64	20.45
+ globalnorm $\rightarrow$ localnorm	12.72	20.46
+ remove time-axis for layernorm	12.71	20.44
+ 250ms future context limit	13.19	21.16

Table 3: *Effect of future context on WER performance.*

Future context	**vid-clean**	**vid-noisy**
250 ms	13.19	21.16
500 ms	13.07	20.81
1000 ms	12.92	20.46
2500 ms	12.80	20.73
5000 ms	12.65	20.44

5.2. Effect of concurrent streams

As discussed in Section 1, processing multiple audio streams in parallel is necessary to achieve higher throughput. Figure 2 shows the effect of increasing the number of concurrent stream on throughput and RTF for a fixed chunk size of 750ms. We see that throughput increases dramatically up to about 30 concurrent streams beyond which it no longer improves. On the other hand, RTF consistently increases as we increase the number of concurrent streams. While any setting with RTF < 1 is considered real-time, as discussed in Section 4.4, lower RTF improves the user-perceived latency.

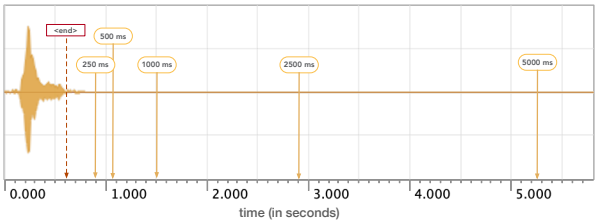

Figure 4: *AM latency when varying the future context size. The transcript of input audio is "yes" and end of this word is marked in the audio with* `<end>`*. Word timestamps generated by greedy decoding for different future context sizes are marked at various locations in the input audio.*

5.3. Effect of chunk size

For streaming speech recognition, we feed the acoustic model with audio in chunks every T milliseconds (where T is the chunk size). The transcript is then generated in an online fashion. Figure 3 shows the effect of increasing the audio chunk size on throughput and latency. We see that even though throughput can be increased by increasing chunk size, the user-perceived latency also increases.

6. Conclusion and Future Work

We present a convolutional online speech recognition system which outperforms two strong baselines in throughput, WER and latency. We also demonstrate the trade-offs in improving one metric over another and how the metrics can be fine-tuned according to the application requirements. In future work we plan to further improve throughput by using 8-bit quantization and techniques like reducing the model depth on demand [29] and weight sparsity [30]. The above techniques may also help to reduce the model size while speeding up inference time. We have also deployed our models on-device, but we leave a full analysis with competitive benchmarks to future work.

7. Acknowledgements

We would like to thank Mahaveer Jain and Duc Le for help in evaluating the baseline models.

8. References

[1] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," *Interspeech 2019*, Sep 2019.

[2] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert, "Wav2letter++: A fast open-source speech recognition system," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6460–6464.

[3] D. S. Park, W. Chan, Y. Zhang *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019.

[4] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," *arXiv preprint arXiv:1805.03294*, 2018.

[5] 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.

[6] 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.

[7] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs rnn in speech applications," *arXiv preprint arXiv:1909.06317*, 2019.

[8] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," *arXiv preprint arXiv:1910.09799*, 2019.

[9] G. Synnaeve, Q. Xu, J. Kahn, E. Grave, T. Likhomanenko, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert, "End-to-end asr: from supervised to semi-supervised learning with modern architectures," *arXiv preprint arXiv:1911.08460*, 2019.

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

[11] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.

[12] Y. Zhang, G. Chen, D. Yu, K. Yaco, S. Khudanpur, and J. Glass, "Highway long short-term memory rnns for distant speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5755–5759.

[13] S. Xue and Z. Yan, "Improving latency-controlled blstm acoustic models for online speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5340–5344.

[14] V. Peddinti, Y. Wang, D. Povey, and S. Khudanpur, "Low latency acoustic modeling using temporal convolution and lstms," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 373–377, 2017.

[15] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[16] M. Jain, K. Schubert, J. Mahadeokar, C.-F. Yeh, K. Kalgaonkar, A. Sriram, C. Fuegen, and M. L. Seltzer, "Rnn-t for latency controlled asr with improved beam search," *arXiv preprint arXiv:1911.01629*, 2019.

[17] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[18] 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.

[19] 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.

[20] L. Kaiser, A. N. Gomez, and F. Chollet, "Depthwise separable convolutions for neural machine translation," *arXiv preprint arXiv:1706.03059*, 2017.

[21] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," *arXiv preprint arXiv:1910.10261*, 2019.

[22] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[23] J. Park, Y. Boo, I. Choi, S. Shin, and W. Sung, "Fully neural network based speech recognition on mobile and embedded devices," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 620–10 630.

[24] D. Le, X. Zhang, W. Zheng, C. Fügen, G. Zweig, and M. L. Seltzer, "From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition," 2019.

[25] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," 2018.

[26] Y. Zhang, G. Chen, D. Yu, K. Yaco, S. Khudanpur, and J. Glass, "Highway long short-term memory rnns for distant speech recognition," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar 2016. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2016.7472780

[27] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Interspeech 2016*, 2016, pp. 2751–2755. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2016-595

[28] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, 11 1992.

[29] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," 2019.

[30] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," 2018.