QUARTZNET: DEEP AUTOMATIC SPEECH RECOGNITION WITH 1D TIME-CHANNEL SEPARABLE CONVOLUTIONS

Samuel Kriman[†⋆] *Stanislav Beliaev*[‡*] *Boris Ginsburg* *Jocelyn Huang*
Oleksii Kuchaiev *Vitaly Lavrukhin* *Ryan Leary* *Jason Li* *Yang Zhang*

NVIDIA, USA
[†]Univ. of Illinois Urbana-Champaign, [‡]High School of Economics, Univ. of Saint Petersburg

ABSTRACT

We propose a new end-to-end neural acoustic model for automatic speech recognition. The model is composed of multiple blocks with residual connections between them. Each block consists of one or more modules with 1D time-channel separable convolutional layers, batch normalization, and ReLU layers. It is trained with CTC loss. The proposed network achieves near state-of-the-art accuracy on LibriSpeech and Wall Street Journal, while having fewer parameters than all competing models. We also demonstrate that this model can be effectively fine-tuned on new datasets.

Index Terms— Automatic speech recognition, convolutional networks, time-channel separable convolution, depthwise separable convolution

1. INTRODUCTION

In the last few years, end-to-end (E2E) neural networks (NN) have achieved new state-of-the-art (SOTA) results on many automatic speech recognition (ASR) tasks. Such models replace the traditional multi-component ASR system with a single, end-to-end trained NN which directly predicts character sequences and therefore greatly simplify training, fine-tuning and inference. The latest E2E models also have very good accuracy, but this often comes at the cost of increasingly large models with high computational and memory requirements.

The motivation of this work is to build an ASR model that achieves SOTA-level accuracy, while utilizing significantly fewer parameters and less compute power. Smaller models offer multiple advantages: (1) they are faster to train, (2) they are more feasible to deploy on hardware with limited compute and memory, and (3) they have higher inference throughput.

We achieve this goal by building a very deep NN with 1D time-channel separable (TCS) convolutions. This new network reaches near-SOTA word error rate (WER) on LibriSpeech [1] (see Table 4) and WSJ [2] (see Table 7) datasets with fewer than 20 million parameters, compared to previous end-to-end ASR designs which typically have over 100 million parameters. We have released the source code and pre-trained models in the NeMo toolkit [3].[1]

2. RELATED WORK

There has been a lot of work done in exploring compact network architectures and on investigating the trade-off between accuracy and size of neural networks, such as SqueezeNet [4], ShuffleNet [5], and EfficientNet [6]. Our approach is directly related to MobileNets [7, 8] and Xception [9], which uses depthwise separable convolutions [10]. Each depthwise separable convolution module is made up of two parts: a depthwise convolutional layer and a pointwise convolutional layer. Depthwise convolutions apply a single filter per input channel (input depth). Pointwise convolutions are 1×1 convolutions, used to create a linear combination of the outputs of the depthwise layer. BatchNorm and ReLU are applied to the outputs of both layers.

Hannun et al [11] applied a similar approach to ASR. They introduced an encoder-decoder model with time-depth separable (TDS) convolutions. The TDS model operates on data in time-frequency-channels ($T \times w \times c$) format, where T is the number of time-steps, w is the input width and c is the number of channels. The basic TDS block is composed of a 2D convolutional block with $k \times 1$ convolutions over $(T \times w)$, and a fully-connected block, consisting of two 1×1 pointwise convolutions operating on $(w \cdot c)$ channels interleaved with layer-norm layers. In contrast, in our work we operate on data in time-channel format ($T \times c$) and completely decouple the time and channel-wise parts of convolution. TDS block has $k \times c^2 + 2 \times (w \cdot c)^2$ parameters, while QuartzNet model has $k \times c + c^2$ parameters, which allows for a dramatic reduction in model size while still achieving good WER.

Another very small ASR model was introduced by Han et al [12], which uses multiple parallel streams of self-attention with dilated, factorized, although not separable, 1D convolutions. The parallel streams capture multiple resolutions of speech frames from the input by using different dilation rates

*Work was conducted while S.Kriman and S.Beliaev were at NVIDIA

[1] https://github.com/NVIDIA/NeMo

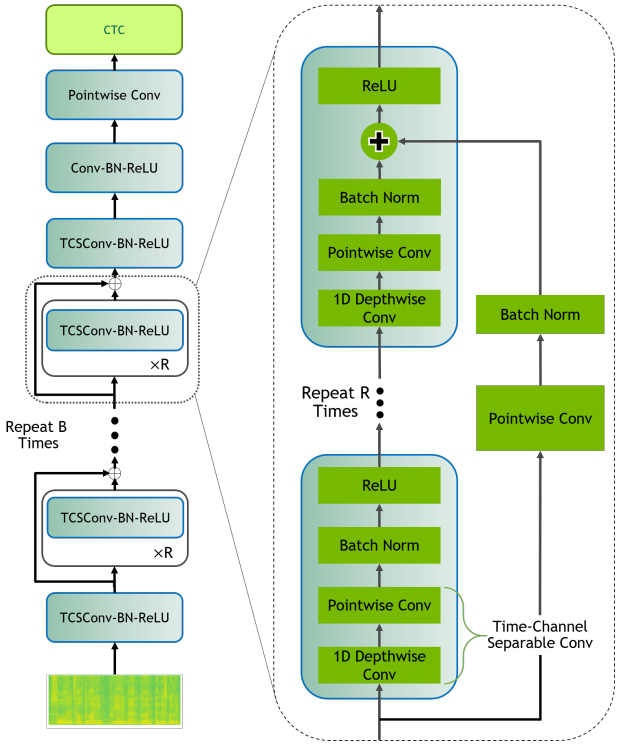

Fig. 1. QuartzNet BxR architecture

per stream, and the results of the individual streams are concatenated into a final embedding. The best model has five streams with dilation rates 1-2-3-4-5.

3. MODEL ARCHITECTURE

3.1. Basic model

QuartzNet's design is based on the Jasper [13] architecture, which is a convolutional model trained with Connectionist Temporal Classification (CTC) loss [14]. The main novelty in QuartzNet's architecture is that we replaced the 1D convolutions with 1D time-channel separable convolutions, an implementation of depthwise separable convolutions. 1D time-channel separable convolutions can be separated into a 1D depthwise convolutional layer with kernel length K that operates on each channel individually but **across K time frames** and a pointwise convolutional layer that operates on each time frame independently but **across all channels**.

QuartzNet models have the following structure: they start with a 1D time-channel separable convolutional layer C_1 followed by a sequence of blocks (see Figure.1). Each block B_i is repeated S_i times and has residual connections between blocks. Each block B_i consists of the same base modules repeated R_i times and contains four layers: 1) K-sized depthwise convolutional layer with c_{out} channels, 2) a pointwise convolution, 3) a normalization layer, and 4) ReLU. The last part of the model consists of one additional time-channel separable convolution (C_2), and two 1D convolutional layers (C_3, C_4). The C_1 layer has a stride of 2, and C_4 layer has a dilation of 2.

Table 1. QuartzNet Architecture. The model starts with a conv layer C_1 followed by a sequence of 5 groups of blocks. Blocks in the group are identical, each block B_k consists of **R** time-channel separable **K**-sized convolutional modules with **C** output channels. Each block is repeated **S** times. The model has 3 additional conv layers (C_2, C_3, C_4) at the end.

Block	R	K	C	S		
				5x5	10x5	15x5
C_1	1	33	256	1	1	1
B_1	5	33	256	1	2	3
B_2	5	39	256	1	2	3
B_3	5	51	512	1	2	3
B_4	5	63	512	1	2	3
B_5	5	75	512	1	2	3
C_2	1	87	512	1	1	1
C_3	1	1	1024	1	1	1
C_4	1	1	$\|\|$labels$\|\|$	1	1	1
Params, M				6.7	12.8	18.9

Table 1 describes the QuartzNet-5x5, 10x5 and 15x5 models. There are five unique blocks across these models: B_1 - B_5. The different models repeat the blocks a different number of times, represented by S_i. QuartzNet-5x5 ($B_1 - B_2 - B_3 - B_4 - B_5$) has each group of blocks repeated 1 time, QuartzNet-10x5 ($B_1 - B_1 - B_2 - B_2 - ... - B_5 - B_5$) - repeated 2 times, and QuartzNet-15x5 ($B_1 - B_1 - B_1 - ... - B_5 - B_5 - B_5$) - repeated 3 times.

The depthwise convolution is applied independently for each channel, so it contributes a relatively small portion of the total number of weights. A regular 1D convolutional layer with kernel size K, c_{in} input channels, and c_{out} output channels has $K \times c_{in} \times c_{out}$ weights. The time-channel separable convolutions use $K \times c_{in} + c_{in} \times c_{out}$ weights split into $K \times c_{in}$ weights for the depthwise layer and $c_{in} \times c_{out}$ for the pointwise layer. This allows us to use much wider kernels, roughly 3 times larger than kernels used in wav2letter [15] or Jasper [13] models (see Table.1). We experimented with four types of normalization: batch normalization [16], layer normalization [17], instance normalization [18], and group normalization [19], and found that models with batch normalization have most stable training and give the best WER.

QuartzNet model is highly scalable, and its accuracy improves with the number of layers (see Table 2).

3.2. Pointwise convolutions with groups

The total number of weights for a time-channel separable convolution block is $K \times c_{in} + c_{in} \times c_{out}$ weights. Since K is

Table 2. QuartzNet models with different depth trained on LibriSpeech for 300 epochs, greedy WER (%).

Model	dev-clean	dev-other	Params, M
5x5	5.39	15.69	6.7
10x5	4.14	12.33	12.8
15x5	3.98	11.58	18.9

generally several times smaller than c_{out}, most weights are concentrated in the pointwise convolution part. In order to further reduce the number of parameters, we explore using group convolutions for this layer. We also added a group shuffle layer to increase cross-group interchange [5] (see Figure 2).

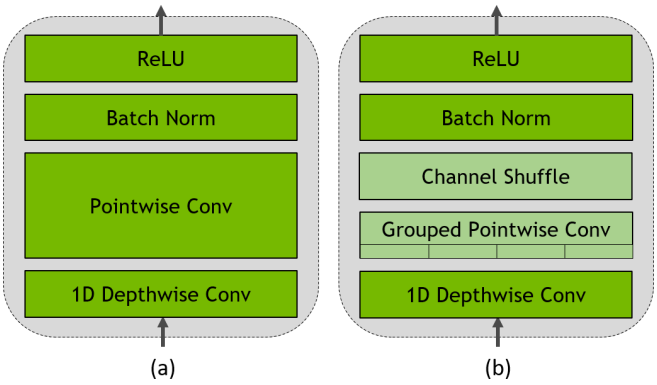

Fig. 2. (a) Time-channel separable 1D convolutional module (b) Time-channel separable 1D convolutional module with groups and shuffle

Using groups allows us to significantly reduce the number of weights at the cost of some accuracy. Table 3. shows the trade-off between accuracy and number of parameters for group amounts one, two, and four, evaluated on LibriSpeech.

Table 3. QuartzNet-15x5 with grouped convolutions trained on LibriSpeech for 300 epochs, greedy WER (%)

# Groups	dev-clean	dev-other	Params, M
1	3.98	11.58	18.9
2	4.29	12.52	12.1
4	4.51	13.48	8.70

4. EXPERIMENTS

We evaluate QuartzNet's performance on LibriSpeech and WSJ datasets. We additionally experiment with a transfer learning showcasing how a QuartzNet model trained with LibriSpeech and Common Voice [20] can be fine-tuned on a smaller amount of audio data, the WSJ dataset, to achieve better performance than training from scratch.

4.1. LibriSpeech

Our best results on the LibriSpeech dataset are achieved with the QuartzNet-15x5 model, consisting of 15 blocks with 5 convolutional modules per block (see Table 1). By combining our network with independently trained language models (i.e., n-gram language models and Transformer-XL (T-XL) [21]) we got WER comparable to the current SOTA (see Table 4).

The model with time-channel separable convolutions is much smaller than a model with regular convolutions and is less prone to over-fitting, so we use only data augmentation and weight decay for regularization during training. We experimented with SpecAugment [22], SpecCutout, and speed perturbation [23]. We achieved the best results with 10% speed perturbation combined with Cutout [24] which randomly cuts small rectangles out of the spectrogram. The models are trained using NovoGrad optimizer [25] with a cosine annealing learning rate policy. We also found that learning rate warmup helps stabilize early training.

Table 4. LibriSpeech results, WER (%)

Model	Augment	LM	Test clean	Test other	Params,M
wav2letter++ [26]	speed perturb	ConvLM	3.26	10.47	208
LAS [22]	SpecAugment	RNN	2.5	5.8	360
TDS Conv [11]	dropout+	-	5.36	15.64	37
	label smooth	4-gram	4.21	11.87	
		ConvLM	3.28	9.84	
MSSA[12]	speed perturb	4-gram	2.93	8.32	23
		4-LSTM	2.20	5.82	
JasperDR-10x5[13]	SpecAugment+	-	4.32	11.82	333
	speed perturb	6-gram	3.24	8.76	
		T-XL	2.84	7.84	
QuartzNet 15x5	SpecCutout+	-	3.90	11.28	19
	speed perturb	6-gram	2.96	8.07	
		T-XL	2.69	7.25	

The training of the 15x5 model for 400 epochs took ≈ 5 days on one DGX1 server with 8 Tesla V100 GPUs with a batch size of 32 per GPU. In order to decrease the memory footprint as well as training time, we used mixed-precision training [27]. We reduced the training time to just over four hours by scaling training to SuperPod with 32 DGX2 nodes with larger number of epochs and with an increased global batch of 16K (see Table 5).[2]

4.2. Wall Street Journal

We trained a smaller QuartzNet-5x3 model on the open vocabulary task of the Wall Street Journal dataset [2]. We used train-si284 set for training, nov93-dev for validation, and nov92-eval for testing. The QuartzNet-5x3 model (see Table

[2]Training even longer (3000 epochs) improved greedy WER on test-clean to 3.87% and on test-other to 10.61%.

Table 5. QuartzNet-15x5: large batch training on LibriSpeech, time to train (hours) and greedy WER (%).

Batch	Epochs	Time, h	Dev		Test	
			clean	other	clean	other
256	400	122	3.83	11.08	3.90	11.28
16K	1500	4.3	3.71	10.78	4.04	11.06

6) was trained for 1200 epochs with batch size 32 per GPU, data augmentation (10% speed perturbation, SpecCutout) and dropout of 0.2 using NovoGrad optimizer ($\beta_1 = 0.95$, $\beta_2 = 0.5$) with 1000 steps of learning rate warmup, a learning rate of 0.05, and weight decay 0.001.

We used 2 external language models during inference: 4-gram (beam size=2048, alpha=3.5, beta=1.5) and Transformer-XL (T-XL). Both language models were constructed using only the official LM data of WSJ.

Table 6. QuartzNet-5x3 for WSJ. The model has the same layers C_1, C_2, C_3, C_4 as QuartzNet-15x5, but the middle part consists of only five blocks, each of which is repeated three times.

Block	R	K	C
C_1	1	33	256
B_1	3	63	512
B_2	3	63	512
B_3	3	75	512
B_4	3	75	512
B_5	3	75	512
C_2	1	87	512
C_3	1	1	1024
C_4	1	1	$\|$labels$\|$

We used following end-to-end models trained on standard speech features[3] for comparison (see Table 7):
1) RNN-CTC [29] - CTC model with 5 bidirectional LSTM layers, 500 cells in each layer;
2) ResCNN-LAS [30]: Listen-Attend-Spell model with deep residual convLSTM encoder and LSTM decoder + label smoothing;
3) Wav2Letter++ [28] - CTC model with 1D convolutional layers and instance norm.

4.3. Transfer Learning

As our model is smaller than other models, we were interested in how well it could learn to generalize to data from various sources, especially if the amount of target speech is much smaller than the training data. Our setup consists of training QuartzNet 15x5 on a combination of LibriSpeech [1] and Mozilla's Common Voice [20][4] datasets, and then fine-tuning this trained model on the 80 hour WSJ dataset. Table 8 shows the WER achieved on LibriSpeech prior to fine-tuning and the result on WSJ after fine-tuning.

Table 7. QuartzNet-5x3, WSJ, WER(%)

Model	LM	93-test	92-eval	Params, M
RNN-CTC[29]	3-gram	-	8.7	26.5
ResCNN-LAS[30]	3-gram	9.7	6.7	6.6
Wav2Letter++[28]	4-gram	9.5	5.6	17
	convLM	7.5	4.1	
QuartzNet-5x3	4-gram	8.1	5.8	6.4
	T-XL	7.0	4.5	

Table 8. QuartzNet15x5 transfer learning. The model was pre-trained of LibriSpeech-train and Mozillas Common Voice datasets, and fine-tuned on the 80-hour WSJ dataset. The model was evaluated on LibriSpeech and WSJ, WER(%).

LM	LibriSpeech		WSJ	
	test-clean	test-other	93-test	92-eval
-	4.19	10.98	8.97	6.37
4-gram	3.21	8.04	5.57	3.51
T-XL	2.96	7.53	4.82	2.99

5. CONCLUSIONS AND FUTURE DIRECTIONS

We introduced a new end-to-end speech recognition model, based on deep neural network with 1D time-channel separable convolutional layers. The model showed close to state-of-the art performance on Wall Street Journal and on LibriSpeech while being significantly smaller than all other end-to-end systems with similar accuracy. The small model footprint opens new possibility for speech recognition on mobile and embedded devices.

This work described a CTC-based model, but we are exploring models where the QuartzNet encoder is combined with attention-based decoders.

6. REFERENCES

[1] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *ICASSP*, 2015, pp. 5206–5210.

[2] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal based CSR corpus," in *Proceedings of the workshop on Speech and Natural Language*. ACL, 1992, pp. 357–362.

[3] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin,

[3]Note, that wav2letter++ [28] with trainable front-end and convLM has even better WER: 6.8% for nov93-test, and 3.5% for nov92-dev. Here, we consider only models with a standard mel-filterbanks front-end.

[4]We used the validated set of Common Voice, ver. en_1087h_2019-06-12.

J. Cook, et al., "NeMo: a toolkit for building AI applications using neural modules," *arXiv:1909.09577*, 2019.

[4] F.N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W.J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model," *arXiv:1602.07360*, 2016.

[5] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," *CVPR*, 2018.

[6] M. Tan and Q.V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *ICML*, 2019.

[7] 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:1704.04861*, 2017.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," *CVPR*, 2018.

[9] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.

[10] V. Vanhoucke, "Learning visual representations at scale," *ICLR*, 2014.

[11] A. Hannun, A. Lee, Q. Xu, and R. Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," in *Proc. Interspeech 2019*, 2019, pp. 3785–3789.

[12] K.J. Han, R. Prieto, K. Wu, and T. Ma, "State-of-the-art speech recognition using multi-stream self-attention with dilated 1D convolutions," *arXiv:1910.00716*, 2019.

[13] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J.M. Cohen, H. Nguyen, and R.T. Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *arXiv:1904.03288*, 2019.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[15] V. Liptchinsky, G. Synnaeve, and R. Collobert, "Letter-based speech recognition with gated convnets," *arXiv:1712.09444*, 2017.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.

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

[18] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv:1607.08022*, 2016.

[19] Y. Wu and K. He, "Group normalization," *Lecture Notes in Computer Science*, p. 319, 2018.

[20] Mozilla, "Common Voice," `https://voice.mozilla.org/en`, 2019.

[21] Z. Dai, Z. Yang, Y. Yang, J.G. Carbonell, Q.V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," *arXiv:1901.02860*, 2019.

[22] D. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E.D. Cubuk, and Q.V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *arXiv:1904.08779*, 2019.

[23] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," *Interspeech*, 2015.

[24] T. Devries and G.W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv:1708.04552*, 2017.

[25] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, H. Nguyen, and Cohen J. M., "Stochastic gradient methods with layer-wise adaptive moments for training of deep networks," *arXiv:1905.11286*, 2019.

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

[27] P. Micikevicius, S. Narang, J. Alben, et al., "Mixed precision training," *arXiv:1710.03740*, 2017.

[28] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," *arXiv:1806.07098*, 2018.

[29] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," *ICML*, 2014.

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