Shrinking Bigfoot: Reducing wav2vec 2.0 footprint

Zilun Peng[1] **Akshay Budhkar**[1] **Ilana Tuil**[2] **Jason Levy**[2]
Parinaz Sobhani[1] **Raphael Cohen**[2] **Jumana Nassour**[2]
[1]Georgian.io [2]Zoominfo
`{zilun.peng, akshay, parinaz}@georgian.io`
`{ilana.tuil, jason.levy}@zoominfo.com`
`{raphael.cohen, jumana.nassour}@zoominfo.com`

Abstract

Wav2vec 2.0 is a state-of-the-art speech recognition model which maps speech audio waveforms into latent representations. The largest version of wav2vec 2.0 contains 317 million parameters. Hence, the inference latency of wav2vec 2.0 will be a bottleneck in production, leading to high costs and a significant environmental footprint. To improve wav2vec's applicability to a production setting, we explore multiple model compression methods borrowed from the domain of large language models. Using a teacher-student approach, we distilled the knowledge from the original wav2vec 2.0 model into a student model, which is 2 times faster, 4.8 times smaller than the original model. More importantly, the student model is 2 times more energy efficient than the original model in terms of CO_2 emission. This increase in performance is accomplished with only a 7% degradation in word error rate (WER). Our quantized model is 3.6 times smaller than the original model, with only a 0.1% degradation in WER. To the best of our knowledge, this is the first work that compresses wav2vec 2.0.

1 Introduction

Speech recognition technologies enhance people's lives through a wide range of applications such as virtual assistants, home automation, and real-time transcription and captioning. Neural network-based speech recognition models achieve superior performances, but successfully training them requires a lot of labeled data (Amodei et al., 2016).

Wav2vec 2.0 (Baevski et al., 2020) is a framework that combines self-supervised masking pre-training with a tuning step that achieves impressive results with little labeled data. With only 10 minutes of labeled data, wav2vec 2.0 achieves word error rates (WER) of 4.8% and 8.2 % on LibriSpeech (Panayotov et al., 2015) dev-clean and dev-other datasets, respectively. Moreover, Wav2vec 2.0 achieves state-of-the-art performances on the LibriSpeech benchmark while using a minimal amount of labeled data. Self-supervised learning enables wav2vec 2.0 to learn speech representations efficiently without much labeled data. This allows easily training an end-to-end automatic speech recognition (ASR) system for a low resource language[1] or for a specific domain. The largest version of wav2vec 2.0 has 317 million parameters, which makes real-time inference inefficient in production.

Language modeling is a central research question in natural language processing (NLP). In recent years massive LM architectures, based on novel pre-training approaches, revolutionized NLP (Devlin et al., 2019; Howard and Ruder, 2018). The BERT (Devlin et al., 2019) architecture of masked pre-training and transformers is similar to wav2vec2.0. To improve wav2vec's applicability to a production setting, we explore multiple model compression methods borrowed from the domain of massive LMs.

A significant drawback of massive LM models is high resource consumption. The negative impact of such models, beyond the costs for organizations using this technology in production, is cause for concern (Strubell et al., 2019; Bender et al., 2021). Various compression methods have been developed to overcome the problem, such as (Sanh et al., 2020). To compress wav2vec 2.0, we explore quantization and knowledge distillation methods. Our quantized model is 3.6 times smaller than the original model, with a 0.1% loss in WER, making it easy to deploy in a production environment. Our student model is at least 2 times faster on both CPU and GPU and 4.8 times smaller than the original model, with a 7% loss in WER through knowledge distillation.

Since the knowledge distilled model is smaller

[1]For example, an Arabic ASR model `https://huggingface.co/elgeish/wav2vec2-large-xlsr-53-arabic`

than the original model, it is more energy efficient. In Section 7, we estimated that the distilled model could potentially save 21 tons of CO_2 equivalents (comparing to the original model) if running inferences on 54 millions of hours of speech data.

Our work makes two key contributions: 1. This research is the first work that compresses the state-of-the-art wav2vec 2.0 speech recognition model to the best of our knowledge. 2. Our distilled student model has a faster inference speed and makes wav2vec 2.0 more cost-efficient and environmental friendly.

2 Related work

In this section, we relate research works done on model compression in ASR systems. Then we describe compression methods applied in language modeling, which we wish to explore for compressing wav2vec.

2.1 Model compression in ASR systems

Several works have been done on compressing end-to-end ASR systems to make them more applicable to production settings. The compression method varies depending on the ASR architecture used. Some of the methods result in approximately 2-3% reduction in accuracy (Mehrotra et al., 2020; Li et al., 2019; Mori et al., 2018), while others result in an improved accuracy (Winata et al., 2020; Kurata and Audhkhasi, 2018). Mehrotra *et al.* (Mehrotra et al., 2020) use reinforcement learning and low rank factorization to compress a system composed of an encoder-attention-decoder with unidirectional LSTMs, while Kurata *et al.* (Kurata and Audhkhasi, 2018) and Takashima *et al.* (Takashima et al., 2018) employ knowledge distillation to train a UniLSTM model from a BiLSTM one. Winata *et al.* (Winata et al., 2019) applied low rank factorization to compress LSTMs. Mori *et al.* (Mori et al., 2018) start with "Deep Speech 2" architecture, which consists of both convolution layers and bidirectional gated recurrent unit layers (GRU), reduce the number of parameters using Tensor-Train decomposition on the weight matrix of the GRUs (TT-GRU).

Some attempts have been made to compress transformer-based ASR systems, like the work done to share parameters across different layers by incorporating additional features related to the topic and the speaker (Li et al., 2019), leading to less than a two-point decrease in accuracy. Another work proposed using a low-rank transformer (LRT) to reduce the number of parameters and boost the speed of training and inference (Winata et al., 2020). The proposed LRT model improved accuracy without using a language model or acoustic data in addition to the improvement in size and speed.

2.2 Classic compression methods in language modeling

Knowledge distillation, quantization, and pruning are few techniques employed in research for compressing language models.

Knowledge distillation involves the use of a teacher model (or models) to improve the performance of another model (student model). When used for compression, the student model's size is significantly smaller than that of the teacher model.

Knowledge distillation in ASR is used both to improve accuracy (Futami et al., 2020; Gao et al., 2020; Mori et al., 2018), and to make the model more applicable to production by reducing its size (Kurata and Audhkhasi, 2018; Takashima et al., 2018), see subsection 2.1. Futami *et al.* (Futami et al., 2020) use knowledge distillation to generate more syntactically or semantically likely results by providing a left context seq2seq ASR system (student model) with soft labels of context to the right. The authors produce soft labels by using a pre-trained BERT model (teacher model). Gao *et al.* (Gao et al., 2020) use multiple teacher models to train a student ASR model to improve ASR accuracy jointly. Moriya *et al.* (Moriya et al., 2020) improve the accuracy by adding another term to the loss function called self-distillation (SD), which comes from incorporating the teacher model.

BERT (Devlin et al., 2019) is one of the most prominent language models in NLP. Like wav2vec, it is based on the transformer model, but it uses only encoders without decoders. DistilBERT (Sanh et al., 2020) is a distilled version of BERT, with a 40% decrease in model size, a 60% increase in speed, and a 97% preservation of language understanding capabilities. Hence, it would be interesting to apply a similar distillation technique to wav2vec. TinyBERT (Jiao et al., 2019) and MobileBERT (Sun et al., 2020) also applied knowledge distillation to compress BERT.

Quantization involves reducing the number of bits used to represent weights. Normally, networks use 32-bit floating-point weights. Because neural networks have millions of such weights, quantizing these weights can make neural networks more energy efficient (Dally, 2015). Several works have been published on quantization, reducing weights to 8-bit (Vanhoucke et al., 2011), 4-bit (Choi et al., 2018), 2-bit (Hwang and Sung, 2014), and 1-bit (Courbariaux et al., 2015) integers. Weights are static and can be quantized through scaling and shifting, but this is not true for activations. Activations vary depending on input samples and can be quantized through static or dynamic quantization. Static quantization learns scaling and shifting of activations through a batch of samples, while dynamic quantization scales and shifts activations in each forward pass. There are two popular software packages for quantization: Distiller (Zmora et al., 2019), and PyTorch (Paszke et al., 2019). Distiller simulates quantization; hence, weights are still represented using 32-bit floating-point numbers but are restricted to integer values. Since we are interested in studying the efficiency of quantizing wav2vec 2.0, we decided not to explore Distiller for quantization. In section 5, we discuss the pros and cons of using PyTorch for quantizing wav2vec 2.0.

Pruning is executed by removing small weights that are likely redundant and can be removed without much loss in accuracy (Han et al., 2015). Han *et al.* (Han et al., 2016) proposed deep compression, an efficient compression pipeline, which first prunes the neural network model and then quantizes the remaining effective weights. We conducted a preliminary study on pruning using the Distiller (Zmora et al., 2019) sensitivity pruner. We set the sensitivity for each convolution layer to 0.1. We set the sensitivity in layers 3 to 16 of the transformer to 0.3 and layers 17 to 23 to 0.4. We pruned weights smaller than the product of the sensitivity and standard deviation of weights on that layer. The pruned model has a 23% sparsity while maintaining the same WER as the original model. We, however, didn't observe any significant improvement in the inference speed.

3 Wav2vec

Wav2vec 2.0 (Baevski et al., 2020) achieved a breakthrough in ASR by adopting the masked pretraining method employed in the massive language model BERT. BERT masks a few words in each training sentence, and the model learns by attempting to fill the gaps. Instead of masking words, wav2vec 2.0 masks parts of the audio representation and lets the transformer network fill in the gaps. This self-supervised setup makes it possible to learn mainly from unlabeled data while only fine-tuning on a small labeled dataset. The wav2vec 2.0 model comprises two significant components that we consider for compression: the CNN layers and the transformer layers.

In the following sections, we explore how compressing a fine-tuned version of wav2vec 2.0[2] using knowledge distillation and quantization affects the size of the model and its inference speed.

4 Knowledge distillation for wav2vec 2.0

We based our distillation on the knowledge distillation used by Sanh *et al.* to compress BERT (Devlin et al., 2019) to DistilBERT (Sanh et al., 2020). They initialized their student models by taking alternating layers from a well-performing larger model and scheduled the learning rate by first increasing, then gradually decreasing it.

Since it is possible to reduce either the CNN layer or the transformer layers of wav2vec 2.0, we decided to reduce the transformer ones since wav2vec 2.0 has only 7 CNN layers, which does not constitute a substantial computational bottleneck compared to transformer layers, see Table 3.

4.1 Knowledge distillation loss

The classic knowledge distillation approach (Hinton et al., 2015) focuses on transferring knowledge between classification neural networks, where both the teacher and the student models output probability distributions over M labels. In our case, the output is latent representations of the input speech audio waveform. We project these representations into probability distributions over M tokens, where a token could map to a character or word boundary. Let T and S be the output probability distributions of the teacher and the student model, respectively. Inspired by (Hinton et al., 2015), we define the knowledge distillation loss to be the Kullback–Leibler (KL) divergence between T and S:

$$L_{distill}(T, S) = \frac{1}{N} \sum_{0 \leq i < N} \sum_{0 \leq j < M} T_{ij} \log \left(\frac{S_{ij}}{T_{ij}} \right) \tag{1}$$

[2]We used wav2vec_big_960h as our base/original model. (https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_big_960h.pt)

The dimension of T and S is $N \times M$, where N is the length of the representation and M is the number of tokens. The KL divergence signals how far off is the student model's prediction is from its teacher, and the student model learns from the teacher by optimizing this loss.

4.2 The architecture of the student model

The only difference between our student model and the original wav2vec 2.0 model is the number of transformer layers. The largest version of wav2vec 2.0 has seven convolution layers and 24 transformer layers, while our student models have a reduced number of transformer layers. Our smallest student model has four transformer layers. We have also trained student models with 6, 8, 10, 12 transformer layers. See section 7 for a discussion on the trade-off between the number of transformer layers and the performance of the different student models.

4.3 Objective function

Our objective function for knowledge distillation training consists of the knowledge distillation loss and feature penalty (Baevski et al., 2020), $L_{distill} + L_{feature}$. $L_{distill}$, the knowledge distillation loss, is defined in equation 1, and $L_{feature}$ is the L2 norm of the final output of convolution layers and serves as a regularization term for convolution layers. The original wav2vec 2.0 uses $L_{feature}$ during training.

We have tried to use the cosine embedding loss (Sanh et al., 2020) in our objective function, but it didn't help with the student model's performances.

4.4 Training techniques

After trying different optimizers, step size scheduling, objective functions, and other techniques to help the student model learn faster, the two approaches that significantly improved the student model's performance were using DistilBERT's initialization strategy and training on more data.

5 Quantization for wav2vec 2.0

In this section, we discuss how the two quantization techniques available in Pytorch apply to our task of quantizing wav2vec 2.0.

5.1 Static quantization

Static quantization quantizes weights and requires calibration before inference to determine scaling and shifting for quantizing activations. PyTorch does not support quantization for gelu activation (Hendrycks and Gimpel, 2020) (as of March 2021), one of the activations used in the wav2vec 2.0 model. A closed pull request [3] shows they were trying to add the support but suspended it temporarily due to some test case failures. One possible workaround is to dequantize before gelu and re-quantize afterward, but this causes some performance degradation.

5.2 Dynamic quantization

Dynamic quantization only quantizes weights and dynamically quantizes activations in each forward pass. As of March 2021, there are no supports for dynamically quantizing the multi-head attention (MHA) mechanism, which is an essential part of a transformer layer. Related issues remain open in PyTorch [4] and fairseq [5]. As a result, tensors entering MHA must not be quantized. We observed that dynamic quantization wraps weights and biases of MHA with some methods, requiring a costly *linear_unpack* operation to access them during inference. We save weights and biases before inference so that *linear_unpack* is not performed for every sample during inference, hence increasing inference efficiency.

6 Experiments

The wav2vec 2.0 model we are trying to compress was trained with 960 hours of unlabeled data from the LibriSpeech dataset (Panayotov et al., 2015), and then fine-tuned with the labeled version of the same 960 hours. We did not attempt to compress the model before it was fine-tuned. We used a Viterbi decoder to convert the output of the student wav2vec 2.0 model into text. We conducted training on 4 Nvidia V100 GPUs and set the batch size to 1 per GPU. We trained with 16-bit floating-point arithmetic and fixed the seed at 42 in all our runs to make the results reproducible.

For knowledge distillation, we used the Adam optimizer with weight decay. We set $eps = 10^{-6}$, $betas = (0.9, 0.98)$, and the weight decay coefficient to 0.01. We scheduled the learning rate by the number of epochs. The learning rate starts from 2.5×10^{-5} and linearly increases for the first two epochs. After two epochs, we linearly decrease the learning rate in every epoch. Also, we assessed the

[3]https://github.com/pytorch/pytorch/pull/34396
[4]https://github.com/pytorch/pytorch/issues/32764
[5]https://github.com/pytorch/fairseq/issues/1901

effect of using a different number of transformer layers on the target metrics, see Table 2.

For quantization, we used PyTorch's dynamic quantization to quantize linear layers of wav2vec 2.0. Table 1 contains the results of evaluating the different compression methods on LibriSpeech dev-clean.

Our implementation is available at `https://git.io/JmpJM`.

7 Results and discussion

Table 1 shows the performance of the largest version of wav2vec 2.0 compared to the compressed models. The distilled model has a 7% loss in terms of word error rate (WER) compared with the original model. However, the distilled model is at least two times faster than the original model on both CPU and GPU, in addition to being 4.8 times smaller than the original wav2vec 2.0 model. We trained the distilled model for 32 epochs.

Table 1 shows that our distilled model is at least 2 times more efficient than the original model in terms of CO_2 emissions and energy consumption. The distilled model is expected to consume less energy since it is smaller than the original model. Our test set contains 5.4 hours of speech data, and the distilled model emits 0.0019 kg of CO_2 equivalents less than the original model. We roughly estimate the potential saving on a larger dataset: if we have 54 millions of hours of speech data, and inference using the distilled model, then it's possible to save $\sim$21 tons of CO_2 equivalents. This shows the energy efficiency of the distilled model, especially when applying it to a large dataset. We used CodeCarbon (Schmidt et al., 2021) to estimate CO_2 emissions and energy consumption. While CodeCarbon should be able to track CPU energy consumption, for our quantized model (which can only be used on a CPU), our machines were not supported by CodeCarbon (as seen in this pull request[6]), and hence we were not able to report CO_2 emissions for our quantized model. Other tools like carbontracker (Anthony et al., 2020) and experiment-impact-tracker (Henderson et al., 2020) can be used in the future to track both CPU and GPU energy consumption.

Considering the quantized model, we notice that even though the model is 3.6 times smaller than the original model, it is not significantly faster. This discrepancy can be attributed to the need for dequantize tensors entering multi-head attention; see section 5.2 for more details. PyTorch does not support GPU quantization; therefore, we report only CPU inference time for the quantized model. Its significant reduction in size, together with its ability to maintain the original WER, albeit with a slight loss, make the quantized model easily deployable in a production environment.

7.1 Knowledge distillation using different settings

There is a wide range of settings that can be tweaked when distilling a model. We explore some of them to understand their effect on the resulting model and the target metrics.

We conducted some experiments to evaluate the trade-off between the size of a model and its performance when performing knowledge distillation by varying the number of kept transformer layers in the student model. LibriSpeech train-clean-100 was used to train the student models. We trained each student model for ten epochs. We performed inferences on a single GPU. Table 2 reports the results of these experiments. The performance of the original wav2vec 2.0 is shown in the first row. We observe that as the number of transformer layers decreases, the inference time decreases, but WER increases. Thus, as expected, as the model becomes smaller, its inference time improves, but its accuracy degrades.

In section 4.2, we introduced the student wav2vec 2.0 model's architecture, where we mentioned that reducing the number of convolution layers will probably not have much effect on the inference time of the distilled model. We tested this claim by reducing the convolution layers to 4. Table 3 shows the result of this experiment compared to the original setting. GPU inference time is only 0.7 seconds faster in the four convolution layer model, whereas CPU inference time is slower. This behavior is because we replaced convolution layers with max-pooling, and PyTorch has a slower max-pooling operation on the CPU.

DistilBERT initializes layers in the student model by taking alternating layers from a larger model. It's possible to initialize layers in the student model by taking the last few layers from a larger model. Figure 1 shows the effectiveness of DistilBERT's initialization strategy, as the student wav2vec 2.0 model initialized using DistilBERT's approach significantly outperforms the

[6]https://github.com/mlco2/codecarbon/issues/156

Model	#Parameters	Model Size	CPU Inference Time	GPU Inference Time	WER	Emissions as CO_2-equivalents	Energy Consumed
Original	317 M	1262 MB	4433 s	123 s	**2.63%**	0.0032 kg	0.0087 kWh
Distilled	**65 M**	**262 MB**	**1560 s**	**51 s**	9.51%	**0.0013 kg**	**0.0036 kWh**
Quantized	317 M	354 MB	4079 s	N/A	2.75%	N/A	N/A

Table 1: Performances of the original wav2vec 2.0 model, a student model trained using knowledge distillation, and a quantized wav2vec 2.0 model.

Layers	Parameters	Model Size	Inference Time	WER
24	317 M	1262 MB	123 s	2.63%
12	166 M	665 MB	81 s	6.6%
10	141 M	564 MB	75 s	7.59%
8	115 M	464 MB	69 s	10.42%
6	91 M	363 MB	60 s	16.54%

Table 2: Performance of a student wav2vec 2.0 model with different number of transformer layers on GPU.

Layers	GPU Inference Time	CPU Inference Time
7	2.97 s	**161 s**
4	**2.26 s**	387 s

Table 3: Inference time of wav2vec 2.0 model with different number of convolution layers. Original wav2vec 2.0 model has 7 convolution layers.

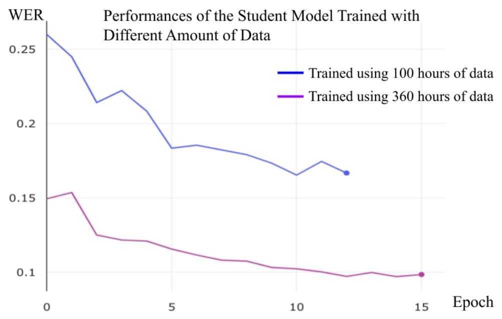

Figure 2: A student wav2vec 2.0 model trained with knowledge distillation using different amount of data.

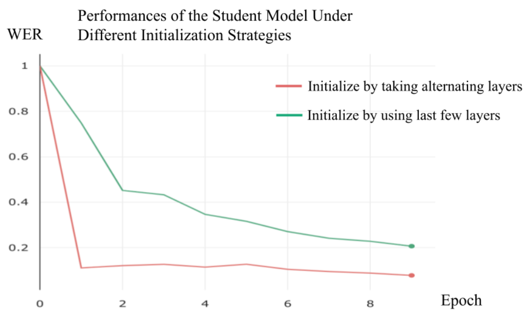

Figure 1: A student wav2vec 2.0 model initialized using different strategies, then trained with knowledge distillation.

other method from the beginning of training. We split the dev-clean dataset into two parts for this experiment, one for training and the other for validation, which we used for reporting WER.

We observed that a student model trained with more data tends to obtain a better WER. As figure 2 shows, the student model trained using 360 hours of data (train-clean-360) achieves better WER than the student model trained with less data (train-clean-100).

8 Conclusion

Jointly training teacher and student model has become a popular technique for compressing ASR systems (Swaminathan et al., 2021; Yu et al., 2020). In contrast, we employed a traditional knowledge distillation approach and it will be an interesting future work direction to investigate why such approach did not achieve great performances for the student model. Despite performance limitations, we found that knowledge distillation is the most effective compression method for wav2vec 2.0 for saving energy by lowering CPU or GPU inference time. Using knowledge distillation, we trained a student wav2vec 2.0 model, which is at least two times faster than the original model on both CPU and GPU. Moreover, the trained student model is also two times more energy efficient than the original model.

Wav2vec models present a valuable technology for democratizing ASR. We explored multiple methods for reducing memory usage and speeding up such models for deployment. More remarkably, our compressed model is environmental friendly during inference. Maintaining low carbon footprint and improving model's performance will be our future research direction.

References

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Awni Y. Hannun, Billy Jun, Tony Han, Patrick LeGresley, Xiangang Li, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Sheng Qian, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Chong Wang, Yi Wang, Zhiqian Wang, Bo Xiao, Yan Xie, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. 2016. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 173–182. JMLR.org.

Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. ArXiv:2007.03051.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3123–3131.

William Dally. 2015. High-performance hardware for machine learning. *NIPS Tutorial*, 2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. 2020. Distilling the knowledge of bert for sequence-to-sequence asr.

Yan Gao, Titouan Parcollet, and Nicholas Lane. 2020. Distilling knowledge from ensembles of acoustic models for joint ctc-attention end-to-end speech recognition.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning.

Dan Hendrycks and Kevin Gimpel. 2020. Gaussian error linear units (gelus).

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Kyuyeon Hwang and Wonyong Sung. 2014. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6. IEEE.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351.

Gakuto Kurata and Kartik Audhkhasi. 2018. Improved knowledge distillation from bi-directional to uni-directional lstm ctc for end-to-end speech recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 411–417. IEEE.

Sheng Li, Dabre Raj, Xugang Lu, Peng Shen, Tatsuya Kawahara, and Hisashi Kawai. 2019. Improving transformer-based speech recognition systems with compressed structure and speech attributes augmentation. In *INTERSPEECH*, pages 4400–4404.

Abhinav Mehrotra, Łukasz Dudziak, Jinsu Yeo, Young-yoon Lee, Ravichander Vipperla, Mohamed S Abdelfattah, Sourav Bhattacharya, Samin Ishtiaq, Alberto Gil CP Ramos, SangJeong Lee, et al. 2020. Iterative compression of end-to-end asr model using automl. *Proc. Interspeech 2020*, pages 3361–3365.

Takuma Mori, Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2018. Compressing end-to-end asr networks by tensor-train decomposition. In *Interspeech*, pages 806–810.

T. Moriya, H. Sato, T. Tanaka, T. Ashihara, R. Masumura, and Y. Shinohara. 2020. Distilling attention weights for ctc-based asr systems. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6894–6898.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5206–5210. IEEE.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 8024–8035.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. 2021. CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic BERT for resource-limited devices. *CoRR*, abs/2004.02984.

Rupak Vignesh Swaminathan, Brian King, Grant P. Strimel, Jasha Droppo, and Athanasios Mouchtaris. 2021. Codert: Distilling encoder representations with co-learning for transducer-based speech recognition. *CoRR*, abs/2106.07734.

Ryoichi Takashima, Sheng Li, and Hisashi Kawai. 2018. An investigation of a knowledge distillation method for CTC acoustic models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 5809–5813. IEEE.

Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. 2011. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*.

Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. 2020. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6144–6148. IEEE.

Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J. Barezi, and Pascale Fung. 2019. On the effectiveness of low-rank matrix factorization for LSTM model compression. *CoRR*, abs/1908.09982.

Jiahui Yu, Wei Han, Anmol Gulati, Chung-Cheng Chiu, Bo Li, Tara N. Sainath, Yonghui Wu, and Ruoming Pang. 2020. Universal ASR: unify and improve streaming ASR with full-context modeling. *CoRR*, abs/2010.06030.

Neta Zmora, Guy Jacob, Lev Zlotnik, Bar Elharar, and Gal Novik. 2019. Neural network distiller: A python package for dnn compression research.