Efficiently Train ASR Models that Memorize Less and Perform Better with Per-core Clipping

Abstract

Gradient clipping plays a vital role in training large-scale automatic speech recognition (ASR) models. It is typically applied to minibatch gradients to prevent gradient explosion, and to the individual sample gradients to mitigate unintended memorization. This work systematically investigates the impact of a specific granularity of gradient clipping, namely per-core clipping (PCC), across training a wide range of ASR models. We empirically demonstrate that PCC can effectively mitigate unintended memorization in ASR models. Surprisingly, we find that PCC positively influences ASR performance metrics, leading to improved convergence rates and reduced word error rates. To avoid tuning the additional hyperparameter introduced by PCC, we further propose a novel variant, adaptive per-core clipping (APCC), for streamlined optimization. Our findings highlight the multifaceted benefits of PCC as a strategy for robust, privacy-forward ASR model training.

Index Terms: gradient clipping, unintended memorization, large ASR models.

1. Introduction

While large neural networks start to exhibit emergent abilities, many generative vision and language models have been found to regurgitate their training data during inference [1, 2, 3, 4]. Therefore, preserving user privacy within the training data of these models has become a critical and widely-studied concern recently. For non-generative automatic speech recognition (ASR) models, there are also a few recent works demonstrating corresponding privacy attacks [5, 6, 7], indicating that speech models are not spared from such privacy risks.

The gold standard of privacy for neural networks is differential privacy (DP) [8], and the workhorse for differentially private large-scale deep learning is differentially private stochastic gradient descent (*i.e.* DP-SGD) [9]. However, the strong privacy of DP-SGD comes to the detriment of computational cost [10] and utility overhead [9], which can be prohibitive in many potential use cases such as training large-scale ASR models. To address the issue, a recent line of work [11, 7] proposes to omit the noise addition operation in DP-SGD and only keep the per-example clipping operation for better utility at the cost of providing only empirical privacy. Surprisingly, per-example clipping on its own is able to significantly mitigate state-of-the-art attacks on ASR models [11, 7].

However, per-example clipping still suffers from computational overhead. The reason is that to conduct per-example clipping, the gradients of all training examples, namely **per-example gradients**, need to be materialized in memory. However, most common deep learning frameworks avoid completely materializing per-example gradients for speed and memory optimization [12] because such materialization places a substantially higher demand on compute resources. As the model size grows larger, such overhead gradually becomes unsustainable. Although there has been a line of work attempting to reduce the memory/compute overhead of per-example clipping, the solutions either incur other trade-offs, such as an extra back-propagation pass [10, 13] or worse utility [14, 15], or do not completely alleviate the overhead [12].

To bypass the issue, we make a key observation that almost all large ASR models are trained with data parallelism [16]. In data parallelism, a mini-batch of training examples is sharded across a few compute cores (*e.g.* GPUs/TPUs). Each compute core runs forward- and backward-propagation on its own data shard to compute the average gradient of the shard, and then all the compute cores synchronize to obtain the average mini-batch gradient. This implies, even in non-private training setups, each core needs to materialize its own average gradient per training step, namely per-core gradient, before aggregating across cores. Therefore, if we clip the per-core gradients instead of per-example gradients, it only incurs negligible memory/compute overhead.

The idea of per-core clipping (PCC) can be viewed as a special case of micro-batch clipping [17, 18] in model training with data parallelism. Traditional micro-batch clipping is also designed to reduce the compute/memory overhead of DP-SGD at the expense of a lower signal-to-noise ratio. PCC, by leveraging the fact that per-core gradients always get materialized in training with data parallelism, almost completely eliminates the compute/memory overhead.

However, a surprising observation from our experiments is that **PCC can improve the performance of ASR models**. Concretely, we conducted an extensive evaluation of PCC across several ASR models with different architectures and datasets, and observed faster convergence rate and lower WER in most of them. Our conjecture for the improvement is that per-core clipping acts as an implicit regularization that prevents the extreme outliers from slowing down the convergence or deviating the model from a better local minima. We leave a thorough investigation into the factors contributing to the observed improvement to future work.

PCC introduces one hyperparameter, the clipping bound, which can require tuning for best results. To address this, we propose a hyperparameter-free variant of per-core clipping, namely adaptive per-core clipping (APCC). By using the minimum L2 norm across the per-core gradients in each iteration as the clipping bound, APCC removes the extra hyperparameter and even further improves the performance metrics.

In summary, we make the following contributions:

- We propose per-core clipping, a variant of gradient clipping that suppresses unintended memorization in ASR mod-

Algorithm 1 Pseudocode for minibatch-SGD with PCC/APCC and data parallelism (changes for PCC in blue, and APCC in red). $\bar{g}$ represents per-core gradients. $\hat{g}$ represents clipped per-core gradients. g represents mini-batch gradients.

Input: initial parameters w_0, loss function $\mathcal{L}$, training data $\mathcal{D}$, #iterations T, per-core batch size B, #compute cores C, learning rate r, clipping bound b.

for $t = 1, 2, \ldots T$ **do**
  $\{d_j^t\}_{j \in \{1,\ldots,BC\}} \leftarrow \mathcal{D}$ ▷ sample mini-batch from $\mathcal{D}$
  for $c = 1, 2, \ldots C$ **parallelly do**
    The cth core loads its data shard $\{d_j^t\}_{j \in \{B(c-1)+1,\ldots,Bc\}}$
    $\bar{g}_t^c = \nabla\mathcal{L}(w_{t-1}, \{d_j^t\}_{j \in \{B(c-1)+1,\ldots,Bc\}})$ ▷ forward/backward propagation on each compute core
    $\hat{g}_t^c = \frac{\min(||\bar{g}_t^c||_2, b)}{||\bar{g}_t^c||_2} \cdot \bar{g}_t^c$ ▷ per-core clipping
  $b_t = \min_{c \in \{1,\ldots,C\}} ||\bar{g}_t^c||$
  for $c = 1, 2, \ldots C$ **parallelly do**
    $\hat{g}_t^c = \frac{b_t}{||\bar{g}_t^c||_2} \cdot \bar{g}_t^c$ ▷ adaptive per-core clipping
  $g^t = \sum_{c \in \{1,\ldots,C\}} \hat{g}_t^c$ ▷ cross-core aggregation
  $w_t = w_{t-1} - r \cdot g^t$ ▷ update the model parameters

els with negligible compute overhead. We showcase that per-core clipping can effectively mitigate unintended memorization on a Conformer [19] model fine-tuned on LibriSpeech [20].

- We perform a comprehensive empirical assessment of PCC across a diverse set models, and observe performance and convergence rate enhancements.
- We propose adaptive per-core clipping, a variant of PCC releasing the burden of extra hyperparameter-tuning.

2. Mitigating Memorization via PCC

In this section, we motivate and describe the design of PCC, and show its empirical privacy advantage under a SOTA attack [7].

2.1. Reducing DP-SGD's Overheads for Empirical Privacy

As mentioned in Section 1, DP-SGD [9] has been a workhorse for large-scale differentially private deep learning since its proposal in 2016. Compared to other DP deep learning algorithms such as DP-FTRL [21] and DP-MF [22, 23, 24] which can achieve better privacy-utility trade-offs by using correlated noise across iterations, the main advantage of DP-SGD is that it is stateless, resulting in a smaller compute overhead.

Compared to its non-private counterpart, DP-SGD still implicitly incurs conspicuous compute/memory and utility overheads. The utility overhead mainly stems from the random noise addition in DP-SGD. [25] One solution to this is to remove the noise addition operation, and only keep the per-example clipping (PEC) operation in DP-SGD. While such training no longer satisfies any meaningful DP, a recent line of work [1, 11, 7] has shown that models trained in this manner tend to match the utility of baseline models while being empirically less prone to memorization.

However, the PEC approach still incurs a compute overhead. [12] The PEC operation requires materializing per-example gradients, which adds a high memory overhead on the training devices. Although there have been a few techniques [10, 14, 13, 15] proposed to avoid/reduce the overhead, they incur other trade-offs, such as an extra round of back-propagation, or worse utility.

2.2. Per-core Clipping

To overcome the compute overhead of PEC, we make a key observation that nowadays, many large ASR models are trained with data parallelism [16]. Consider such a setting (shown in Algorithm 1) where, for each training step, a mini-batch of training examples are sharded on multiple compute cores. To obtain a mini-batch gradient, aggregated gradients (called *per-core gradients*) are computed at each core before undergoing a cross-core aggregation. Thus, an operation of Per-core Clipping (PCC), i.e., clipping the per-core gradients, does not incur any memory overhead to the training pipeline.

While the memory advantage of PCC over PEC increases with the number of training examples in each core (*i.e.,* the per-core batch size), the effect of mitigating memorization can reduce as a result. Think about the two extreme cases: 1) When per-core batch size is 1, PCC reduces to PEC and provides exactly the same trade-offs; and 2) when per-core batch size equals the mini-batch size (i.e., no data parallelism), PCC reduces to purely a learning rate scaling operation without affecting the minibatch gradients.

2.3. Measuring Unintended Memorization

Secret Sharer & Exposure: To study the empirical privacy benefits of PCC, we use the Secret Sharer framework [1], and follow the methodology specific to ASR models in [7]. Concretely, we use a WaveNet Text-to-Speech (TTS) engine [26] to generate 4x-sped-up utterances, and insert them into the training set of the ASR models. The transcripts fed to the TTS engine are a combination of random words of length 7 following Wang *et al.* [7]. These utterances as called canaries, and the goal is to design them to be out-of-distribution enough from the regular training examples so that a model is not able to generalize to them without seeing them during training. If an ASR model demonstrates unexpectedly high accuracy in transcribing canary utterances, this suggests strong evidence that the model has engaged in unintended memorization of training data rather than developing robust generalization capabilities. Moreover, such memorization can make the model susceptible to leakage about its training data via privacy attacks.

In this work, we measure the trained ASR model's character error rate (CER) of both the canaries and utterances drawn from the same distribution but unseen during training (*i.e.,* a heldout set). If the model does exceptionally well on the canaries com-

Table 1: *Exposure of canaries by CER with different insertion frequencies. Mean and standard deviation are calculated from 20 canaries.* ***Bold*** *highlights best.*

Canary #Insertion	1	2	4	8	16
Baseline	4.8 ± 3.7	11.0 ± 3.8	13.0 ± 3.5	13.2 ± 3.4	13.5 ± 2.9
PCC@2.5	$\mathbf{1.0} \pm 0.0$	$\mathbf{1.0} \pm 0.0$	$\mathbf{1.0} \pm 0.0$	$\mathbf{1.5} \pm 2.3$	$\mathbf{2.1} \pm 3.2$
APCC	1.7 ± 1.8	1.7 ± 1.3	1.3 ± 1.5	2.2 ± 1.5	2.5 ± 2.3

Table 2: *Best WER on LibriSpeech within 20K fine-tuning steps. Mean and standard deviation are calculated across 3 runs.* ***Bold*** *highlights best.*

Test set	Baseline	PCC	APCC
test-clean	1.93 ± 0.04	1.87 ± 0.05	$\mathbf{1.85} \pm 0.07$
test-other	3.99 ± 0.01	$\mathbf{3.80} \pm 0.04$	$\mathbf{3.80} \pm 0.05$

pared to heldout set, this indicates the model has memorized the canaries. This intuition can be formally captured by exposure, a metric used to measure such memorization. The higher the exposure of a canary, the stronger it has been memorization.

Definition 1 (Exposure [1]) *Given a canary c, a model $\mathcal{M}$, and examples in a holdout set r_i, the exposure of c is*

$$\textbf{\textit{exposure}}_{\mathcal{M}}(c, \{r_i\}) = \log_2 |\{r_i\}| - \log_2 rank_{\mathcal{M}}(c, \{r_i\}),$$

where $|\{r_i\}|$ is the size of the holdout set, and $rank_{\mathcal{M}}(c, \{r_i\})$ is the rank of canary c among r_i in terms of a metric of interest, e.g., loss, or CER.

Experimental Setup: We use the 600M Conformer XL [27], a state-of-the-art ASR model architecture, for our memorization analysis. The encoder is pre-trained on the LibriLight dataset [28] for 1 million steps using the BEST-RQ [29] self-supervised training technique. Next, we attach a decoder to the model, and fine-tune the complete model on the LibriSpeech dataset [20] for 20,000 steps. The canaries are inserted during the fine-tuning phase, and the training approach mirrors [7]. The per-core batch size used is 4, and 128 cores are used to train each model.

Evaluation Results: We train Conformer XL with and without PCC, and the first 2 rows in Table 1 summarize the privacy evaluation results. The clipping bound 2.5 is selected based on WER on LibriSpeech test-other by grid searching $\{1, 2.5, 5, 10, 100\}$ on Conformer M [19], a 20x smaller model with a similar architecture. We use 2.5 for all the following experiments without further tuning the hyper-parameter. We first observe that for baseline training, even canaries appearing only once in the training set have an average exposure of 4.8, while for per-core clipping, exposure is at its lower bound (i.e., no detection of unintended memorization) until 4 repetitions of the canary in the training set. We also notice that the standard deviation of exposure for the model trained with PCC is 0.0 for canaries appearing at most 4 times in the training set. To understand this better, we manually inspected the decoding output for such canaries by the PCC model, and found that the model outputs empty for the inserted fast canaries, perhaps an ideal behavior when encountering indecipherable utterances.

2.4. Adaptive Per-core Clipping

Training ASR models can encompass a variety of architectures and datasets, and hence lead to different gradient norms during training. As a result, deploying PCC can involve tuning the clipping bound. To alleviate such burden, we devise a variant called Adaptive PCC (APCC), which adaptively uses the minimum L2-norm among all the per-core gradients as the clipping bound for each training step, as shown in details in Algorithm 1.

We ran the same exposure analysis as in Section 2.3 for APCC, and the results are summarized in the 3rd row of Table 1. We observe that APCC also shows much lower exposure compared to the baseline model. However, the exposure numbers are higher than PCC with clipping bound 2.5. We conjecture this behavior could be attributed to the fact that the minimum per-core gradient norm is used without any privacy protection. However, we leave this investigation for future work.

3. Better ASR performance via PCC

Since PCC provides better empirical privacy with no extra compute overhead, a natural question to ask is whether it causes any regression in ASR performance like DP-SGD. In this section, we evaluate PCC across a variety of model architectures and datasets. To our surprise, we find that PCC consistently improves their ASR performance.

3.1. Case Study 1: Conformer on LibriSpeech

We first evaluate the model used in Section 2, the 600M Conformer XL, and provide the results in Table 2. We observe that on LibriSpeech test-other, after adding PCC, the average WER is improved from 3.99 to 3.80, a 4.8% relative improvement. APCC achieves the same average WER of 3.80. On LibriSpeech test-clean, the WER is also improved using PCC/APCC by 3.1%/4.1% relative, respectively. Note that the standard deviations are small for all settings.

3.2. Case Study 2: E2E ASR Model for Voice Search

For our second case study, we evaluate the effectiveness of per-core clipping on a large-scale voice search task.

Model Architecture: The models for this case study follow the architecture described in Prabhavalkar *et al.* [30]. Concretely, the models are hybrid autoregressive transducers (HAT) [31], comprised of an encoder, a prediction network, and a joint network. The encoder is comprised of a convolutional subsampling block, followed by a series of 16 conformer blocks whose order of the convolution and multi-headed self-attention layers are exchanged. The prediction network is a V^2 embedding prediction network [32]. The joint network is a linear layer followed by tanh activation. More details about the model architecture can be found in [30].

Training and Evaluation Sets: The training set is en-us utterances extracted from Google voice search traffic. The majority of utterances are pseudo-labeled using a teacher model [33], while a small portion is anonymized and human-transcribed, following Google AI principles [34]. The test set is composed of 6 datasets, a voice search test set (VS), corresponding to the

Table 3: *Conformer for Voice Search Benchmark w/ or w/o PCC.* ***Bold*** *highlights better.*

Model	Baseline						PCC					
	VS	RM	RN	RP	RQ	RY	VS	RM	RN	RP	RQ	RY
A0	4.0	13.0	14.8	37.7	20.8	23.9	**3.8**	**12.7**	**12.8**	**37.2**	**20.5**	**23.4**
A1	3.8	14.2	15.6	37.0	19.9	23.2	**3.7**	**12.2**	**15.5**	**36.7**	**19.4**	**22.9**
A2	3.7	10.3	16.1	33.1	16.2	**20.1**	**3.6**	**9.9**	**14.1**	**31.5**	**15.1**	20.3

Table 4: *Modular Domain Adaptation for Voice Search Benchmark w/ or w/o PCC.* ***Bold*** *highlights better.*

Model	Baseline						PCC					
	VS	RM	RN	RP	RQ	RY	VS	RM	RN	RP	RQ	RY
MDA (en-us)	5.2	4.1	15.9	11.3	25.0	26.6	**5.1**	**4.0**	**15.8**	**11.1**	**24.9**	**26.5**
MDA (fr-fr)	9.5	-	-	-	-	-	**9.2**	-	-	-	-	-

"head" of the utterance distribution, and 5 rare word test sets containing rare words from different domains including maps (RM), news (RN), Google Play (RP), search queries (RQ), and YouTube (RY), corresponding to the "tail" of the utterance distribution. The rare word test sets are generated following the recipe described in [35].

Evaluation Results: We train the described model with and without PCC. The evaluation results are summarized in Table 3, where model A0 corresponds to B0 in [30], A1 corresponds to E7 [30], and A2 is A1 augmented with text injection [36]. 512 compute cores are used to train the model and the per-core batch size is 8 for A0 and A1, and 16 for A2. We can observe that across all the architectures, the models trained with PCC almost always achieve a better WER on both voice search set and the rare word sets. On A0, PCC achieves 5%/4.1% average relative improvement in WER on VS/rare word test sets. On A1, PCC achieves 2.6%/3.9% average relative improvement in WER on VS/rare word test sets. On A2, PCC achieves 2.7%/5.4% average relative improvement in WER on VS/rare word test sets.

3.3. Case Study 3: Modular Domain Adaptation for Streaming Voice Search

Now, we evaluate on the voice search task described in Section 3.2 above, but using the model architecture following [37].

Model Architecture: We train streaming Conformer transducers [19] following the recipe in [37]. The encoder of the transducer consists of 7 causal Conformer blocks followed by 10 non-causal Conformer blocks [38, 39]. There are two separate hybrid autoregressive transducer (HAT) decoders [31] for the causal and non-causal encoders. More details about the model architecture can be found in [30].

Training and Evaluation Sets: The model is first trained on YouTube (YT) data to obtain a backbone model. Then, the backbone model is fine-tuned on voice search (VS) dataset as described in Case Study 2. Speaker tags [40] were used during training to improve performance. Note that we have two languages in Table 4. When training the en-us (fr-fr) model, both YT and VS dataset are mainly composed of en-us (fr-fr) utterances. We do not have fr-fr rare word test sets.

Evaluation Results: We train the described model with and without PCC. 128 cores are used to train each model, and the per-core batch size is 32. We can observe from the results in Table 4 that after adding PCC, the WER is consistently improved across both languages and all the test sets. The en-us model's performance on VS gets a 2.1% relative improvement, and a 1.1% average relative improvement on the rare word datasets. The fr-fr model's performance on VS sees a 3.2% relative improvement.

3.4. Discussion: ASR Improvement by PCC

While PCC consistently improves ASR quality, the reasons behind this improvement remain open for investigation. We hypothesize that PCC reduces the impact of abnormally large gradients typically caused by extreme outliers (e.g., noisy utterances, foreign language content, music). A crucial question is whether this improvement comes at the cost of performance on tail examples. However, our findings in Case Studies 1 & 2 suggest PCC does not sacrifice performance on tail examples. Rather, we observe enhanced performance on rare word test sets. This suggests a regularization-like dynamic, where PCC suppresses overfitting to extreme outliers in the training set while improving generalization to unseen tail examples. To verify this, we measured WER on the training set [20] and generalization error (*i.e.* the difference of WER between test and training set) of the models outlined in Section 2.3. The results in Table 5 support the hypothesis: PCC exhibits typical regularization behavior: higher training loss and smaller generalization error. This indicates PCC acts as an implicit regularization method. We leave investigation into the connection between PCC and traditional regularization methods to future work.

Table 5: *Generalization Gap w/ or w/o PCC.*

	WER on train-clean	Generalization Gap
Baseline	1.17 ± 0.18	3.00 ± 0.16
PCC@2.5	1.34 ± 0.18	2.63 ± 0.15

4. Conclusion & Future Directions

In this work, we explored the challenges associated with mitigating unintended memorization and maintaining computational efficiency during the training of large-scale ASR models. We proposed PCC, which can effectively mitigate such memorization with negligible computational overhead. Surprisingly, we also observed PCC to improve the performance of a variety of ASR models, lowering WER. We also introduced APCC to alleviate extra hyperparameter tuning.

5. References

[1] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *USENIX Security Symposium*, 2019.

[2] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *USENIX Security Symposium*, 2021.

[3] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," in *USENIX Security Symposium*, 2023.

[4] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," *arXiv preprint arXiv:2311.17035*, 2023.

[5] E. Amid, O. D. Thakkar, A. Narayanan, R. Mathews, and F. Beaufays, "Extracting targeted training data from ASR models, and how to mitigate it," in *Interspeech*, 2022.

[6] M. Jagielski, O. Thakkar, and L. Wang, "Noise masking attacks and defenses for pretrained speech models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[7] L. Wang, O. Thakkar, and R. Mathews, "Unintended memorization in large asr models, and how to mitigate it," in *ICASSP*, 2024.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Third Theory of Cryptography Conference (TCC)*, 2006.

[9] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *The ACM SIGSAC conference on computer and communications security*, 2016.

[10] X. Li, F. Tramer, P. Liang, and T. Hashimoto, "Large language models can be strong differentially private learners," *arXiv preprint arXiv:2110.05679*, 2021.

[11] W. R. Huang, S. Chien, O. D. Thakkar, and R. Mathews, "Detecting unintended memorization in language-model-fused ASR," in *Interspeech*, 2022.

[12] P. Subramani, N. Vadivelu, and G. Kamath, "Enabling fast differentially private sgd via just-in-time compilation and vectorization," *Neural Information Processing Systems (NeurIPS)*, 2021.

[13] Z. Bu, J. Mao, and S. Xu, "Scalable and efficient training of large convolutional neural networks with differential privacy," *NeurIPS*, 2022.

[14] Z. Bu, S. Gopi, J. Kulkarni, Y. T. Lee, H. Shen, and U. Tantipongpipat, "Fast and memory efficient differentially private-sgd via jl projections," *NeurIPS*, 2021.

[15] J. He, X. Li, D. Yu, H. Zhang, J. Kulkarni, Y. T. Lee, A. Backurs, N. Yu, and J. Bian, "Exploring the limits of differentially private deep learning with group-wise clipping," *arXiv:2212.01539*.

[16] J. Chong, G. Friedland, A. Janin, N. Morgan, and C. Oei, "Opportunities and challenges of parallelizing speech recognition," in *The USENIX conference on Hot topics in parallelism. HotPar*, 2010.

[17] H. B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz, "A general approach to adding differential privacy to iterative training procedures," *arXiv:1812.06210*.

[18] N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. G. Thakurta, "How to dp-fy ml: A practical guide to machine learning with differential privacy," *Journal of Artificial Intelligence Research*, 2023.

[19] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Interspeech*, 2020.

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *ICASSP*, 2015.

[21] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu, "Practical and private (deep) learning without sampling or shuffling," in *International Conference on Machine Learning (ICML)*, 2021.

[22] S. Denisov, H. B. McMahan, J. Rush, A. Smith, and A. Guha Thakurta, "Improved differential privacy for sgd via optimal private linear operators on adaptive streams," *NeurIPS*, 2022.

[23] C. A. Choquette-Choo, H. B. McMahan, K. Rush, and A. Thakurta, "Multi-epoch matrix factorization mechanisms for private machine learning," *arXiv preprint arXiv:2211.06530*.

[24] C. A. Choquette-Choo, A. Ganesh, R. McKenna, H. B. McMahan, K. Rush, A. G. Thakurta, and Z. Xu, "(amplified) banded matrix factorization: A unified approach to private training," *arXiv:2306.08153*.

[25] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Annual Symposium on Foundations of Computer Science*, 2014.

[26] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *ICML*, 2018.

[27] Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang, Q. V. Le, and Y. Wu, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504*.

[28] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen *et al.*, "Libri-light: A benchmark for asr with limited or no supervision," in *ICASSP*, 2020.

[29] C.-C. Chiu, J. Qin, Y. Zhang, J. Yu, and Y. Wu, "Self-supervised learning with random-projection quantizer for speech recognition," in *ICML*, 2022.

[30] R. Prabhavalkar, Z. Meng, W. Wang, A. Stooke, X. Cai, Y. He, A. Narayanan, D. Hwang, T. Sainath, and P. Moreno, "Extreme encoder output frame rate reduction: Improving computational latencies of large end-to-end models," in *ICASSP*, 2024.

[31] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (hat)," in *ICASSP*, 2020.

[32] R. Botros, T. N. Sainath, R. David, E. Guzman, W. Li, and Y. He, "Tied & reduced rnn-t decoder," *arXiv:2109.07513*.

[33] D. Hwang, K. C. Sim, Z. Huo, and T. Strohman, "Pseudo label is better than human label," *arXiv preprint arXiv:2203.12668*, 2022.

[34] "Google AI Principles," https://blog.google/technology/ai/ai-principles/, accessed 2023-09-01.

[35] C. Peyser, S. Mavandadi, T. N. Sainath, J. Apfel, R. Pang, and S. Kumar, "Improving tail performance of a deliberation e2e asr model using a large text corpus," *arXiv:2008.10491*.

[36] C. Peyser, Z. Meng, K. Hu, R. Prabhavalkar, A. Rosenberg, T. N. Sainath, M. Picheny, and K. Cho, "Improving joint speech-text representations without alignment," *arXiv:2308.06125*.

[37] Q. Li, B. Li, D. Hwang, T. N. Sainath, and P. M. Mengibar, "Modular domain adaptation for conformer-based streaming asr," *arXiv:2305.13408*.

[38] A. Narayanan, T. N. Sainath, R. Pang, J. Yu, C.-C. Chiu, R. Prabhavalkar, E. Variani, and T. Strohman, "Cascaded encoders for unifying streaming and non-streaming asr," in *ICASSP*, 2021.

[39] T. N. Sainath, Y. He, A. Narayanan, R. Botros, W. Wang, D. Qiu, C.-C. Chiu, R. Prabhavalkar, A. Gruenstein, A. Gulati *et al.*, "Improving the latency and quality of cascaded encoders," in *ICASSP*, 2022.

[40] G. P. Arumugam, S.-Y. Chang, T. N. S. R. Prabhavalkar, Q. Wang, and S. Bijwadia, "Improved long-form speech recognition by jointly modeling the primary and non-primary speakers," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023.