INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Federated Learning with Dynamic Transformer for Text to Speech

Zhenhou Hong, Jianzong Wang, Xiaoyang Qu, Jie Liu, Chendong Zhao, Jing Xiao

Ping An Technology (Shenzhen) Co., Ltd., China

jzwang@188.com

Abstract

Text to speech (TTS) is a crucial task for user interaction, but TTS model training relies on a sizable set of high-quality original datasets. Due to privacy and security issues, the original datasets are usually unavailable directly. Recently, federated learning proposes a popular distributed machine learning paradigm with an enhanced privacy protection mechanism. It offers a practical and secure framework for data owners to collaborate with others, thus obtaining a better global model trained on the larger dataset. However, due to the high complexity of transformer models, the convergence process becomes slow and unstable in the federated learning setting. Besides, the transformer model trained in federated learning is costly communication and limited computational speed on clients, impeding its popularity. To deal with these challenges, we propose the federated dynamic transformer. On the one hand, the performance is greatly improved comparing with the federated transformer, approaching centralize-trained Transformer-TTS when increasing clients number. On the other hand, it achieves faster and more stable convergence in the training phase and significantly reduces communication time. Experiments on the LJSpeech dataset also strongly prove our method's advantage.

Index Terms: Text-to-Speech, Speech Synthesis, Transformer models, Dynamic Growing, Federated Learning

1. Introduction

Text to speech (TTS) [1–3] synthesis is an active research area. Despite decades of research, generating natural speech from text is still a challenging task. With the development of voice technology in recent years, various voice services and applications have new advances, affecting people's daily lives. Over time, different technologies have taken over the field. There are now feasible ways to produce natural prosody with high audio fidelity using a much-simplified voice building pipeline [4–7]. Recently, many variants of transformer models such as FastSpeech [8], Transformer-TTS [9], AlignTTS [1], MultiSpeech [10] have shown great promises in the TTS task. However, such transformer models typically require sizable high-quality training data to achieve a good performance. Nevertheless, limitations like data privacy, liability, and regulatory concerns may make it difficult for clients to collaborate with other data owners for centralized aggregation training.

To tackle this problem, federated learning [11–14] has presented as a paradigm that allows clients to train a shared global model collaboratively without data breaches. When training models in the federated learning setting, participating clients do not upload their local data to the central server; instead, a central aggregator coordinates the optimization procedure among the clients [15, 16]. At each iteration of this procedure, clients compute gradient-based optimization using their local data to update the current model parameters and then upload only these model's updates to the central aggregator. The federated learning framework promises to enhance the model's performance by multipartite data aggregation under clients' absolute privacy conditions.

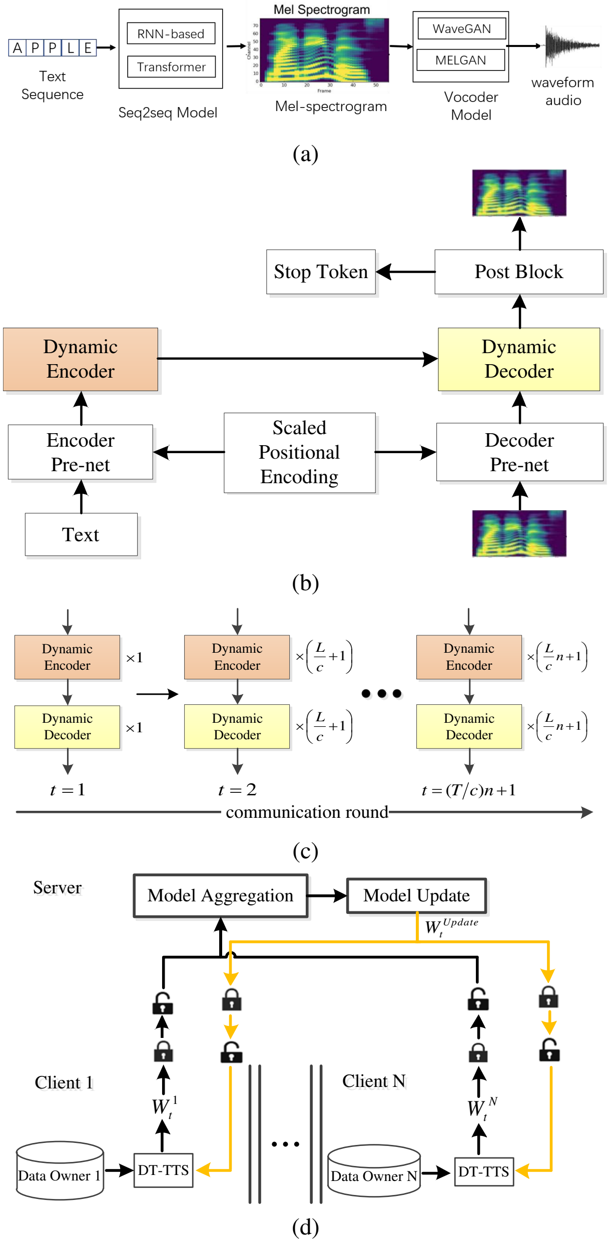
Federated learning has proved to be a compelling framework in different areas, such as optical character recognition (OCR) [17], natural language processing (NLP) [18, 19], Medical Imaging [20] and so on. These show that federated learning has a promising prospect in further practical applications. However, little researches have been done in the TTS area, and especially the model is as large as a transformer. To be specific, the main impediments are as follows: Firstly, all distributed learning methods fatally suffer from the slow and easily unstable convergence problem, which is mainly caused by different clients' non-IID(including quantity skew) [12, 21]. When larger amounts of parameters are contained, the model will be more challenging to converge. Secondly, training such a large model is costly in the federated learning setting. Large amounts of parameters aggravate the computational burden locally and extremely increase the communication time and cost.

To address these issues, we propose the Dynamic Transformer (FedDT-TTS) with faster convergence speed and lower communication cost in the federated learning framework for the TTS task. The key insight is, by changing the layer-wise training in the wake-sleep algorithm [22], we grow both the encoder and decoder dynamically through the training process. This dynamic adding layers mechanism makes the shallow layers in encoder and decoder faster of learning low-level texts or mel features, thus allowing the deeper layers in encoder and decoder to learn high-level text or mel feature information more easily. Moreover, in order to stabilize the convergence, we let the new layer initialize with weights divided by a per-layer normalize constant according to He's initializer [23]. In the meantime, we utilize the Front Layer Normalization (Front-LN) in the FedDT-TTS. Following the residual block in ResNetV2 [24] and its symmetric structure explanation, the Front-LN method puts the layer normalization inside the residual connection and equips it with an additional final-layer normalization before prediction. This position change of layer normalization would greatly help to converge faster and more steadily. Similarly, in the structure of the transformer, we put the layer normalization in front of the Multi-Head Attention and Feed Forward Network (FFN) [25].

Our contributions are as follows:

- We propose the Dynamic Transformer, where encoder and decoder increase layers dynamically. This method enables the model to learn the low-level features better, thus greatly improving transformer models' performance in the federated learning, reducing total training time by 40%.
- We introduce a simple implementation of weight normalization and the Front-LN approach to achieve a faster and more stable convergence process.

http://dx.doi.org/10.21437/Interspeech.2021-2039



(a)

(b)

(c)

(d)

Figure 1: *The diagram of Dynamic Transformer(DT) and Federated Dynamic Transformer(FedDT) for TTS.(a)The pipeline of a typical TTS system;(b)The diagram of our Dynamic Transformer for TTS;(c)The progressive growing of the encoder/deocder block during communication rounds;(d)The diagram of our Federated Dynamic Transformer for TTS. Note that the lock icon means encryption and the unlock icon means decryption.*

2. Proposed Method

2.1. Dynamic Transformer for TTS

As shown in Figure1(a), the typical TTS system comprises two separate modules: a seq2seq model and a vocoder model. The pipeline of the TTS system is shown as follows. First, the text sequence is fed into the seq2seq model, and the output of the seq2seq model is Mel-spectrograms. Then, a vocoder is deployed to convert the spectrogram into time-domain waveforms.

Here, we use Transformer as the seq2seq model because Transformer TTS enables parallel training and provides an opportunity for building long-range dependencies directly. However, as the transformer model's model size hinders the decentralized training and deployment in the mobile devices, we propose Dynamic Transformer (DT). As illustrated in Figure 1(b), we used a dynamic encoder and a dynamic decoder. The details of Text pre-net are described in [9]. To be specific, the Text Pre-net contains an embedding layer and an encoder pre-net, generating a vector for the transformer encoder. Three fully connected layers compose the Mel Pre-net, converting Mel-spectrograms to vectors as well.

The dynamic growing encoder/decoder blocks process is shown in Figure 1(c), where the number l_t of layers is following the communication round t. As an example, when t is equal to a number like 2, then the $l_2 = (L/c + 1)$, where L is the target layers number, c means that L is divided into c parts, and each part has L/c.

In the early stage, the generation of low-level features is substantially more stable because there are less class information and fewer modes. By increasing the layers step by step, we are continuously asking for a much higher-level feature. Another benefit is the reduced training time. With DT, most of the lower-level features are learned at early periods, and considerable performance is often obtained faster in training time. The reason is dynamic transformer being in the Taylor-expansion way: the starting terms account most as low ordering ones, while the subsequential ones can refine the function well. Hence, adding more layers makes the synthesized wave more natural, since it does better in processing spectrogram details.

When adding layers, it occurs a sudden shock to the already well-trained shallow layers. The new layer with randomly initialized weights would make the loss oscillation. So it will let the performance decline to some degree for the first time. To stabilize the training process, we make two changes in model structure and weight normalization, respectively. Firstly, utilize Front-LN in the transformer architecture. We explicitly scale the weights at runtime. To be precise, we set $\hat{w} = w/z$, where w are the weights and z is the per-layer normalization constant from He's initializer [23]. The benefit of doing this dynamically is somehow subtle but relates to the scale-invariance in commonly used adaptive stochastic gradient descent methods such as RMSProp [26] and Adam [27]. These methods normalize a gradient update with its estimated standard deviation, thus making the update independent of the scale of the parameters. As a result, if some parameters have a more extensive dynamic range than others, they will take more time to adjust. These dues to a scenario modern initializers cause, and thus the learning rate may be both too large and too small at the same time.

2.2. Federated Dynamic Transformer for TTS

To ensure the security and privacy of original data, we introduce federated learning to the Text-to-Speech synthesis. As the communication overhead is the bottleneck of the federated learning, we deploy our Dynamic Transformer for TTS model training in a federated setting, as shown in Figure 1(d). Following the classical federated learning optimization method FedAvg [28], in each communication round, firstly, the server randomly choose M clients composing S_t, then send the encrypted current global model's weights to every client in S_t. Each client decrypts it and updates the weights using its local data. Secondly, clients upload their updated weights to the server through the same encrypt-decrypt procedure. After collecting updates, the server would do model aggregation by averaging all clients' updates together, then update the global model with the aggregation value.

Algorithm 1 Federated Dynamic Transformer

```
Input: total communication round T, clients number M, q layers to increase, interval sets of progressive steps $\mathbb{K}$, weights w, transformer model f, the target layers number L, client's local iteration steps I.
Server executes:
for each round t = 1,2,..., T do
    $S_t \leftarrow$ (randomly chosen M clients)
    $k = t * I$
    if ($k \in \mathbb{K}$) and ($l_t < L$) then
        adding layers: $l_t \leftarrow l_t + q$
    end if
    for each client $i \in S_t$ in parallel do
        $w^i_{t+1} \leftarrow$ ClientUpdate$(i, w_t, l_t)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} w^i_{t+1}$
end for

ClientUpdate$(i, w_t, l_t)$:
if adding layers then
    $f(w, l) \leftarrow f(w, l_t)$
end if
for local step $j = 1, .., I$ do
    $w \leftarrow w - \eta \nabla f(w, l)$
end for
return $w$ to server
```

The federated dynamic transformer(FedDT) is described in Algorithm 1. In clients, if adding layers, renew the $f(w, l)$ with $f(w, l_t)$. In the whole training process, the trained layer's weight is keeping after adding the new layer. The benefit of the dynamically growing process when training transformer in the federated learning setting contains two aspects.

First, both encoder and decoder only have shallow layers at the beginning rounds, which massively reduces the communication cost. Even the layer number equals the target layer number in the latter communication rounds, the communication cost in the whole training process is much less than training all layers from beginning to end. Here is a simple analysis of the communication efficiency of FedDT, following the same setting in Algorithm 1. For FedT, assuming the amount of layers in encoder and decoder is N, weights of encoder block and decoder block are W_1 and W_2. Then the total communication cost of FedT is $T(NW_1 + NW_2) = TN(W_1 + W_2)$. For FedDT, communication cost of each round is $\frac{T}{c}\left[(\frac{N}{c}n + 1)W_1 + (\frac{N}{c}n + 1)W_2\right]$, where n is from 0 to $c - \frac{c}{N}$. Because it's an arithmetic sequence, the sum of all rounds is equal to $\frac{T}{2c}\left[(W_1 + W_2) + (NW_1 + NW_2)\right] = \frac{T}{2c}(N + 1)(W_1 + W_2)$. Comparing these two total communication cost, FedDT reduces by $\frac{1}{2c} + \frac{1}{2cN}$ ratio theoretically. If let $N + 1 \approx N$ for simplicity, the reduction ratio is close to $\frac{1}{2c}$.

Second, FedDT's convergence becomes faster and more stable than FedT. The reason is that adding layers progressively can make the model's training begin at a good initial weight, thus converging easily. Similar to the layer-wise training process in the wake-sleep algorithm, when a layer is trained after specific iterations, it gains a well weight distribution. Then add a new layer above it. The trained layer can be seen as a good initial weight to the new layer. While training the new layer, the old layer can be seen as a stable and well-trained feature extractor. Facing the complexity of the federated learning framework helps the training process become more robust.

3. Experiments

3.1. Dataset

All experiments are trained on the LJSpeech data [29]. This dataset consists of 13,100 short audio clips of a single speaker reading passages from 7 non-fiction books. Clips vary in length from 1 to 10 seconds and have a total length of approximately 24 hours. The texts are normalized and inserted with the beginning character (a space) and the end character (a period), e.g. "I am 10 years old" is converted to " i am ten years old.". The LJSpeech dataset is randomly divided into two sets: 12600 samples for training and 500 samples for testing.

Separate the training dataset into five subsets, each having 2520 samples. Each client owns one subset, which has a length of 4.6 hours roughly. Additionally, the test set has a length of almost 1 hour.

3.2. Settings

Model Architecture: The transformer model is following [9]. The DT-TTS and Transformer-TTS configurations are as follows: the hidden size, attention head, FFN filter size are 384, 4, 1536 respectively. These hyper-parameters are also the same as when fitted in federated learning. For the Transformer-TTS, the layers of encoder and decoder are fixed to 6. The inputs for models were all texts and Mel-spectrograms.

Compare Schemes: We analyze our model in two schemes: (i) A standalone case comparing Transformer-TTS and our DT-TTS, both are trained on the entire training dataset in a single worker; (ii) Considering Transformer-TTS in (i) as a baseline, we evaluate both Transformer-TTS and our DT under the federated learning framework.

Evaluation: To quantify models' performances, we use the mean opinion score (MOS) to measure generated voices' qualities. Twenty native English speakers judge each sentence. There are ten males and ten females.

Training Strategy: We experimented with the standalone case with 4 NVIDIA V100 GPUs and trained the FedDT-TTS with 1 NVIDIA V100 GPU on each client. The total communication rounds T is 120, and target layers L for both encoder and decoder are assigned to 6. We set the c equal to 6. Then the L/c is 1, and the T/c is 20, that means we add 1 layer for both the encoder and decoder every 20 communication rounds. ALL training processes utilize a mini-batch Adam optimizer, setting the batch size to 16 for every client. In addition, RSA encryption is used to protect clients' privacy.

3.3. Analysis

The Standalone Case: In Table 1, both models are trained on the entire training dataset. Our DT-TTS converages almost 20% faster than the T-TTS, which trains the transformer model with six layers from beginning to end. From the perspective of MOS, DT-TTS is 4.03 ± 0.11, which is nearly close to T-TTS (4.01 ± 0.07). Our model achieves a comparable performance and reduces the training time appreciably.

In Federated Learning: As a visual illustration in Figure 2, FedT-TTS and FedDT-TTS both improve their performances when the number of clients increases. Obviously, federated

Table 1: *Comparison of Average Convergence Steps and MOS.*

Model	Average Convergence Steps	MOS
T-TTS	697125	4.03±0.11
DT-TTS	539833	4.01±0.07

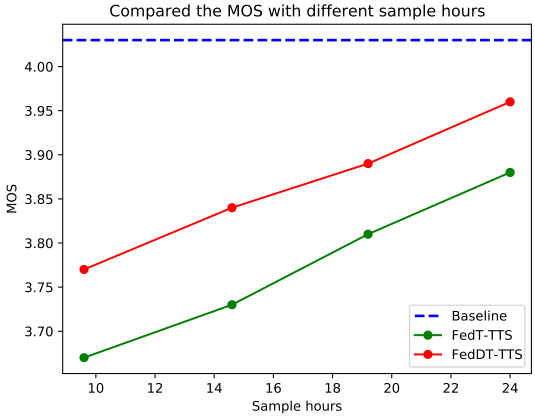


Figure 2: *Models' comparison when increasing sample datas.*

learning greatly helps by collaborating more data from other parties. In Table 2, we detail the MOS comparison between FedT-TTS and FedDT-TTS. We can see that when increasing the client's number, MOS for both models is ascending, our FedDT-TTS is approaching baseline's performance when all clients participating (3.96 to 4.03), especially. Besides, ours outperforms FedT-TTS in terms of client's numbers. On the other hand, ours hugely reduces the total training time by around 20% in different clients number with the superior performance. Especially in a 5-clients situation, FedDT-TTS's time cost is 49.26h, almost 30% reduction compared with FedT-TTS, 66.42h. We can see an almost linear reduction of FedDT-TTS according to the increasing number of clients. This majorly owes to the fewer parameters brought by the dynamic process, reducing training time and communication time. Above all, these significantly demonstrate that ours is more stably trained and time-efficient than the FedT-TTS.

Table 2: *Time cost and MOS comparision between FedT-TTS and FedDT-TTS. "M" is the number of clients.*

Model	M	Time Cost	MOS
T-TTS	-	-	4.03±0.11
FedT-TTS	2	23.54h	3.67±0.11
FedDT-TTS	2	18.35h	**3.77±0.11**
FedT-TTS	3	33.86h	3.73±0.10
FedDT-TTS	3	26.44h	**3.84±0.05**
FedT-TTS	4	48.16h	3.81±0.08
FedDT-TTS	4	37.47h	**3.89±0.07**
FedT-TTS	5	66.42h	3.88±0.06
FedDT-TTS	5	49.26h	**3.96±0.04**

Table 3: *MOS comparision between FedT-TTS and FedDT-TTS "r" is the ratio of spliting with M being 3.*

Model	r of 3 clients	MOS
FedT-TTS	1:1:1	3.73±0.10
FedDT-TTS	1:1:1	**3.84±0.05**
FedT-TTS	1:1:3	3.66±0.11
FedDT-TTS	1:1:3	**3.74±0.09**
FedT-TTS	1:2:4	3.68±0.10
FedDT-TTS	1:2:4	**3.78±0.05**
FedT-TTS	1:1:5	3.63±0.08
FedDT-TTS	1:1:5	**3.72±0.06**
FedT-TTS	1:1:6	3.66±0.07
FedDT-TTS	1:1:6	**3.75±0.04**
FedT-TTS	1:1:8	3.70±0.07
FedDT-TTS	1:1:8	**3.79±0.05**

On Unbalanced Dataset: We also compare the performance of FedT-TTS and FedDT-TTS on an unbalanced dataset scenario, which commonly happens in real federated learning applications. We investigate in a 3-clients setting, resplit the whole training dataset into three subsets according to the different split ratios r, then assign each subset to each client. As the Table 3 shows that on the unbalanced dataset, our FedDT-TTS still outperforms the FedT-TTS. For the MOS of FedDT-TTS, the average improvement is about +0.09 for all split ratios, compared with the FedT-TTS. These results suggest that, with the dynamic adding layers mechanism and Front-LN approach, the FedDT-TTS is more suitable in federated learning, even on the unbalanced dataset.

4. Conclusion

In this paper, we propose the Dynamic Transformer for TTS synthesis. Comparing with T-TTS, ours shows a significant reduction in training time with miniature MOS deducted. Combining with the federated learning framework, we improve the model's performance by collaborating with more data owners under absolute data privacy protection. The MOS of our model keeps increasing while adding clients, approaching the baseline method gradually. Besides, experiments on the LJSpeech datasets show that our FedDT-TTS surpasses FedT-TTS in time efficiency and model performance. On the one hand, the dynamic adding layers process significantly saves client's training time and communication costs. On the other hand, it contributes to improving global convergence robustly. Moreover, experiments on an unbalanced dataset and Mel-spectrograms analysis also prove that our DT-TTS is superior to Transformer-TTS in the federated learning framework.

5. Acknowledgement

This work is supported by National Key Research and Development Program of China under grant No.2018YFB0204403, No.2017YFB1401202 and No.2018YFB1003500. Corresponding author is Jianzong Wang from Ping An Technology (Shenzhen) Co., Ltd.

6. References

[1] Z. Zeng, J. Wang, N. Cheng, T. Xia, and J. Xiao, “Aligntts: Efficient feed-forward text-to-speech system without explicit alignment,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6714–6718.

[2] A. Sun, J. Wang, N. Cheng, H. Peng, Z. Zeng, and J. Xiao, “Graphtts: graph-to-sequence modelling in neural text-to-speech,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6719–6723.

[3] A. Sun, J. Wang, N. Cheng, H. Peng, Z. Zeng, L. Kong, and J. Xiao, “Graphpb: Graphical representations of prosody boundary in speech synthesis,” in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 438–445.

[4] M. Aso, S. Takamichi, and H. Saruwatari, “End-to-end text-to-speech synthesis with unaligned multiple language units based on attention,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 4009–4013.

[5] J. Cho, P. Zelasko, J. Villalba, S. Watanabe, and N. Dehak, “Learning speaker embedding from text-to-speech,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 3256–3260.

[6] Y. Gao, W. Zheng, Z. Yang, T. Köhler, C. Fuegen, and Q. He, “Interactive text-to-speech system via joint style analysis,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 4447–4451.

[7] T. Kenter, M. Sharma, and R. Clark, “Improving the prosody of rnn-based english text-to-speech synthesis by incorporating a BERT model,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 4412–4416.

[8] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, “Fastspeech: Fast, robust and controllable text to speech,” in *Advances in Neural Information Processing Systems*, 2019, pp. 3171–3180.

[9] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural speech synthesis with transformer network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6706–6713.

[10] M. Chen, X. Tan, Y. Ren, J. Xu, H. Sun, S. Zhao, and T. Qin, “Multispeech: Multi-speaker text to speech with transformer,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 4024–4028.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *arXiv preprint arXiv:1912.04977*, 2019.

[13] J. Zhao, X. Zhu, J. Wang, and J. Xiao, “Efficient client contribution evaluation for horizontal federated learning,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3060–3064.

[14] Y. Liu, X. Zhu, J. Wang, and J. Xiao, “A quantitative metric for privacy leakage in federated learning,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3065–3069.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[16] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[17] X. Zhu, J. Wang, Z. Hong, T. Xia, and J. Xiao, “Federated learning of unsegmented chinese text recognition model,” in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 1341–1345.

[18] F. Liu, X. Wu, S. Ge, W. Fan, and Y. Zou, “Federated learning for vision-and-language grounding problems,” in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 11 572–11 579. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/6824

[19] X. Zhu, J. Wang, Z. Hong, and J. Xiao, “Empirical studies of institutional federated learning for natural language processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 625–634.

[20] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, “Secure, privacy-preserving and federated machine learning in medical imaging,” *Nature Machine Intelligence*, pp. 1–7, 2020.

[21] A. Hard, K. Partridge, C. Nguyen, N. Subrahmanya, A. Shah, P. Zhu, I. Lopez-Moreno, and R. Mathews, “Training keyword spotting models on non-iid data with federated learning,” in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, H. Meng, B. Xu, and T. F. Zheng, Eds. ISCA, 2020, pp. 4343–4347.

[22] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *2016 - 14th European Conference Computer Vision , Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, vol. 9908. Springer, 2016, pp. 630–645.

[25] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, and C. Xing, “On layer normalization in the transformer architecture,” in *Proceedings of the 37th International Conference on Machine Learning, 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 524–10 533.

[26] T. Tieleman and G. Hinton, “Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning,” *Technical Report.*, 2017.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, ser. Proceedings of Machine Learning Research, A. Singh and X. J. Zhu, Eds., vol. 54. PMLR, 2017, pp. 1273–1282.

[29] K. Ito *et al.*, “The lj speech dataset,” 2017.