INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Reversible Neural Networks for Memory-Efficient Speaker Verification

Bei Liu, Yanmin Qian[†]

MoE Key Lab of Artificial Intelligence, AI Institute
X-LANCE Lab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{beiliu, yanminqian}@sjtu.edu.cn

Abstract

Training large-scale speaker verification systems on consumer GPUs is difficult due to the memory consumption of the existing networks being proportional to the number of layers. In this paper, a novel family of Reversible Neural Networks (RevNets) is proposed for memory-efficient speaker verification. Specifically, we introduce two types of RevNets, namely partially and fully reversible networks, which alleviate the need to store activations in memory during back-propagation. Consequently, RevNets require nearly constant memory costs as the network depth increases. Experiments on Voxceleb show that RevNets achieve up to **15.7x** memory savings, while maintaining nearly identical parameters and performance when compared to the vanilla ResNets. To our knowledge, this is the first work to investigate memory-efficient training for speaker verification. Our results indicate the potential of reversible networks as a more efficient backbone for resource-limited training scenarios.

Index Terms: speaker verification, memory-efficient training, reversible neural networks

1. Introduction

Speaker verification (SV) is a biometric identification task that involves confirming a person's identity by analyzing the voice characteristics. A typical SV system consists of two components: an embedding extractor and a similarity scorer. The embedding extractor is responsible for extracting the speaker embedding from the variable-length utterances, while the similarity scorer calculates the similarity between the embeddings. Traditionally, i-vector [1] with probabilistic linear discriminant analysis (PLDA) [2] is the most commonly-used method. Currently, deep speaker embedding learning based on neural networks becomes the predominant approach [3, 4, 5, 6, 7].

Recently, the performance of speaker verification systems has been significantly improved as the depth of neural networks increases. For example, [7] presents a depth-first variant of ResNet, substantially increasing the network's depth to 233 layers. Furthermore, [8] makes r-vector deeper and extends the depth of ResNet to 293. Although deeper and larger neural networks have achieved remarkable performance enhancements, the memory consumption of existing networks is directly proportional to the number of layers during training. This characteristic poses a challenge for training large-scale speaker verification systems on consumer GPUs. In general, modern neural networks are trained using back-propagation algorithm [9], which necessitates the storage of intermediate network activations in memory. This procedure incurs a memory cost that scales proportionally with the depth of network, which means that training memory consumption increases linearly with the growth of network depth [10, 11, 12]. On the other hand, graphics processing units (GPUs) have limited memory capacity [13]. To reduce memory usage in large-scale SV systems training, a common way is to decrease the batch size. However, excessively small batch sizes can hinder the accurate estimation of gradients and batch normalization, ultimately hurting the system performance. The effective training of significantly deeper networks, particularly on consumer GPUs, is a critical and challenging task for speaker verification.

[†] corresponding author

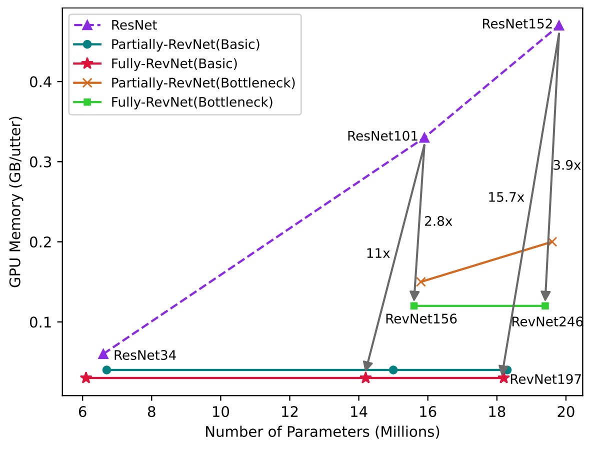

Figure 1: *GPU Memory Usage vs. Parameter Number*

In this paper, we propose a novel family of Reversible Neural Networks (RevNets) for memory-efficient speaker verification. Specifically, two types of RevNets, namely partially and fully reversible networks, are introduced. They comprise a stack of reversible blocks where each layer's activations can be reconstructed from the next layer's ones. This feature enables us to perform back-propagation without the requirement of storing the activations in memory. As a result, RevNets enjoy the benefit of maintaining nearly constant memory usage as the depth of network increases. Experiments on Voxceleb demonstrate that compared to the conventional ResNets, the proposed RevNets achieve a memory savings of up to **15.7x**, while maintaining nearly identical parameters and performance. To our knowledge, this is the first work to explore the feasibility of memory-efficient training for speaker verification. Our results illustrate that reversible networks have the potential to serve as more efficient backbones for resource-constrained training scenarios.

2. Related Work

Deep Speaker Embedding Learning: [14] makes a pioneering effort to apply deep neural networks to speaker verification.

10.21437/Interspeech.2023-844

Subsequently, diverse network architectures have been extensively explored. [15] proposes a time-delay neural network (TDNN) for text-independent speaker verification. Additionally, x-vector [3] and ECAPA-TDNN [5] incorporate several architectural enhancements to improve performance. Meanwhile, the winner of VoxSRC 2019 [4] introduces ResNet as the speaker embedding extractor, which has gained growing popularity in the SV field [16, 17]. Moreover, [18] presents a transformer-based system with self-attention encoder to yield speaker embedding. Plus, [19] builds a pure multilayer perceptron (MLP) network for SV task. Recently, much deeper and larger neural networks have been developed by [7, 8], resulting in remarkable performance improvements.

Reversible Architectures: Reversible networks are a family of architectures which contain a stack of reversible blocks with the ability of recovering the intermediate activations through the inverse functions during back-propagation. They have been successfully adopted in various fields [20, 21, 22, 23, 24, 25, 26]. Specially, [20] proposes a reversible architecture for image classification based on nonlinear independent components estimation (NICE) [27] transformation. [21, 22] introduce reversible U-Net for memory-efficient medical image segmentation. [23] develops reversible 3D convolutional neural networks for snapshot compressive imaging. [24, 25] present reversible transformer for neural machine translation. [26] attempts to reduce the memory costs for speech enhancement using neural ordinary differential equations (NODEs).

3. Reversible Neural Networks

In this section, we firstly introduce the back-propagation algorithm and reversible operators. Then, two different Reversible Neural Networks (RevNets), namely partially and fully RevNets, are proposed for memory-efficient speaker verification.

3.1. Preliminaries

Back-propagation: The back-propagation algorithm [9], also known as reverse-mode automatic differentiation, computes the gradients of the loss function with respect to the neural network's parameters by traversing the computation graph in a backward direction using the chain rule. It is the default training algorithm for modern neural networks, which is adopted in prevalent deep learning frameworks including Tensorflow [10], Pytorch [11] and JAX [12]. Given a computation graph $\mathcal{G}$, we represent the topological ordering of the nodes as $v_1, \ldots, v_l$, where v_l corresponds to the loss function $\mathcal{L}$. Each node can be defined as a function f_i of its parent nodes in $\mathcal{G}$. The back-propagation algorithm can compute the total derivative $d\mathcal{L}/dv_i$ for each node v_i in reverse topological order using the chain rule as follows:

$$\frac{d\mathcal{L}}{dv_i} = \sum_{j \in Child(i)} \left(\frac{\partial f_j}{\partial v_i}\right)^T \frac{d\mathcal{L}}{dv_j} \tag{1}$$

where $Child(i)$ stands for the children of node v_i in $\mathcal{G}$. $\partial f_j / \partial v_i$ is the Jacobian matrix, which calculates the partial derivative of the f_j with respect to the current node v_i.

Due to the Jacobian matrix, the back-propagation algorithm requires the availability of intermediate activations to compute the derivatives with respect to parameters. In general, this is accomplished by caching the intermediate activations in GPU memory for later use in the backward pass, which makes the memory usage linearly dependent on the network depth.

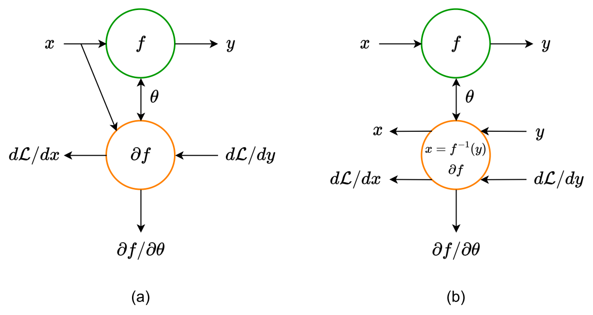

Figure 2: *Comparison between non-reversible operator (a) and reversible operator (b).*

Reversible Operator: Reversible operator refers to a type of transformation with the property of analytic invertibility, implying that the input of the transformation can be reconstructed from its output through an inverse operation. Traditional neural networks are mainly constructed using non-reversible operators. In contrast, a network composed of reversible operators alleviates the need to store intermediate activations in GPU memory during training since they can be recomputed on the fly from the output in the backward pass. Therefore, reversible neural networks effectively decouple the memory usage from the network depth, resulting in substantial memory savings during the training of large-scale networks. Figure 2 schematically depicts the differences between reversible and non-reversible operators. For an inverse function $y = f(x)$, the input x can be recovered by calling $x = f^{-1}(y)$ in the backward pass. Hence, the memory costs can be saved by discarding the intermediate activation x during the forward pass.

In this paper, we investigate memory-efficient training for speaker verification. Specifically, two different Reversible Neural Networks (RevNets), namely partially and fully RevNets, are proposed in the following section.

3.2. Partially Reversible Neural Networks

In this part, we firstly introduce partially reversible neural networks based on ResNet [28]. In the vanilla ResNet, He et al. propose two residual architectures, namely *Basic* block and *Bottleneck* block, as the network building components. Each block consists of a stack of convolution layers, batch normalization (BN) and non-linear activation function (ReLU).

To construct a reversible version of ResNet, the primary challenge is how to transform the non-reversible residual blocks into their reversible counterparts. Inspired by [20], we begin by partitioning the input activation x into x_1 and x_2 evenly along the channel dimension. By utilizing the additive coupling rule, the initial residual block can then be converted into a reversible one as illustrated below.

$$\begin{aligned} y_1 &= x_1 + \mathcal{F}(x_2) \\ y_2 &= x_2 + \mathcal{G}(y_1) \end{aligned} \tag{2}$$

where (x_1, x_2) are inputs and (y_1, y_2) are outputs. $\mathcal{F}$ and $\mathcal{G}$ can be the *Basic* or *Bottleneck* block.

For the reverse process, the input activations can be reconstructed from the output as follows:

$$\begin{aligned} z_1 &= y_1 \\ x_2 &= y_2 - \mathcal{G}(z_1) \\ x_1 &= z_1 - \mathcal{F}(x_2) \end{aligned} \tag{3}$$

Algorithm 1 Back-propagation for Reversible Blocks

```
Input: $(y_1, y_2)$; $(d\mathcal{L}/dy_1, d\mathcal{L}/dy_2)$
Output: $(x_1, x_2)$; $(d\mathcal{L}/dx_1, d\mathcal{L}/dx_2)$; $(d\mathcal{L}/dw_{\mathcal{F}}, d\mathcal{L}/dw_{\mathcal{G}})$
1: $z_1 \leftarrow y_1$;
2: $x_2 \leftarrow y_2 - \mathcal{G}(z_1)$;
3: $x_1 \leftarrow z_1 - \mathcal{F}(x_2)$;
4: $\frac{d\mathcal{L}}{dz_1} \leftarrow \frac{d\mathcal{L}}{dy_1} + \left(\frac{\partial \mathcal{G}}{\partial z_1}\right)^T \frac{d\mathcal{L}}{dy_2}$;
5: $\frac{d\mathcal{L}}{dx_2} \leftarrow \frac{d\mathcal{L}}{dy_2} + \left(\frac{\partial \mathcal{F}}{\partial x_2}\right)^T \frac{d\mathcal{L}}{dz_1}$;
6: $\frac{d\mathcal{L}}{dx_1} \leftarrow \frac{d\mathcal{L}}{dz_1}$;
7: $\frac{d\mathcal{L}}{dw_{\mathcal{F}}} \leftarrow \left(\frac{\partial \mathcal{F}}{\partial w_{\mathcal{F}}}\right)^T \frac{d\mathcal{L}}{dz_1}$;
8: $\frac{d\mathcal{L}}{dw_{\mathcal{G}}} \leftarrow \left(\frac{\partial \mathcal{G}}{\partial w_{\mathcal{G}}}\right)^T \frac{d\mathcal{L}}{dy_2}$;
```

Note that the invertibility property can only hold true when the stride in residual block $\mathcal{F}$ and $\mathcal{G}$ is 1. Otherwise, the layer will discard information, making it non-reversible. The original ResNet architecture includes spatial downsampling at the start of each stage, which is achieved by convolution with a stride of 2. This means that these downsampling layers can not be converted into reversible layers using the above method. Therefore, we retain these non-reversible downsampling layers, yielding partially reversible neural networks that include a few non-reversible layers.

For reversible blocks, we can perform back-propagation without storing intermediate activations. Specifically, given the activations (y_1, y_2) and their total derivatives, we can first reconstruct the input activations (x_1, x_2). Afterward, the total derivative with respect to (x_1, x_2) and any parameters associated with $\mathcal{F}$ and $\mathcal{G}$ can be calculated using Algorithm 1. For non-reversible layers, the explicit storage of activations is mandatory. However, the number of such layers is small. The total memory usage is still independent of the network depth.

3.3. Fully Reversible Neural Networks

As discussed above, the conventional spatially downsampling operations such as strided convolution and max-pooling are inherently non-invertible because they involves altering the spatial dimensionality of input activations. This process leads to the loss of some information, thereby making the operations non-bijective. As a result, the downsampling activations still have to be cached in the backward pass. We call reversible neural networks that contain non-reversible downsampling layers as partially RevNets. In this section, we aim to develop fully reversible neural networks by making the downsampling operation used in the vanilla ResNets invertible.

[29] introduces a pixel squeezing operation for image data. It begins to divide the image into subsquares of shape $C \times 2 \times 2$ per channel, followed by a reshape operation that converts these subsquares to $4C \times 1 \times 1$. The squeezing operation can transform a $C \times H \times W$ tensor into a $4C \times H/2 \times W/2$ shape by effectively trading spatial size for number of channels in an invertible way. Inspired by this idea, we attempt to achieve the downsampling effect by introducing a ratio r to rearrange the feature into a shape of $r^2C \times F/r \times T/r$ for an audio feature with the shape $C \times F \times T$ where C, F and T denote the channel, frequency and time dimension respectively. By replacing the non-reversible downsampling layers with this invertible squeezing operation, fully reversible neural networks are finally obtained.

4. Experimental Setup

4.1. Datasets

We conduct experiments on Voxceleb1&2 [30, 31] datasets. The development set of Voxceleb2 is adopted as training data. The whole Voxceleb1 is used as the evaluation data. Performance is reported on the three official trials: Vox1-O, Vox1-E and Vox1-H. In addition, data augmentation techniques are utilized to increase the diversity of training data. Specifically, we generate extra data samples by adding noise from MUSAN [32] and RIR dataset [33] in an online manner [34]. Meanwhile, the speed of utterances is adjusted by a factor of 0.9 and 1.1 [35], making the number of speakers and utterances tripled.

4.2. Training Settings

During training, we randomly chunk a 200-frame segment from each utterance. Then 80-dimensional Fbank with a window length of 25ms and a shift of 10ms are extracted as the input acoustic feature. AAM-softmax [36] with a margin of 0.2 and a scale of 32 is employed as the loss function. The optimizer is stochastic gradient descent (SGD) with momentum of 0.9 and weight decay of $1e$-4. The learning rate is scheduled by an exponential function which decreases from 0.1 to $1e$-5. The dimension of speaker embedding is set to 256. Both ResNets and RevNets stick to the same training settings.

4.3. Evaluation Protocol

For testing, we use cosine distance as the similarity criterion. Then, the scores are calibrated using adaptive score normalization (AS-Norm) [37] with an imposter cohort size of 600. Equal error rate (EER) is reported for performance measurement.

5. Results and Analysis

5.1. Architectural Specifications

In the experiments, ResNet34, 101 and 152 are adopted as the baseline systems. To align with the parameter number, we elaborately design their corresponding reversible counterparts to make them have similar parameters across different regimes. In Table 1, detailed architectural configurations are listed, including the network depth (layer number), width (channel number in each block for different stages), block type (*Basic* or *Bottleneck*) and parameter number. We propose a series of variants with different configurations for each baseline system. For example, partially RevNet46 and fully RevNet57 are introduced based on *Basic* block with similar parameters to ResNet34. Likewise, four different reversible architectures are developed for ResNet101 and 152 under nearly identical or fewer parameters.

5.2. Performance Comparison

From Table 1, it can be observed that our proposed reversible architectures exhibit a comparable performance to the vanilla ResNets across all models (ResNet34, 101 and 152). Specifically, both partially and fully reversible counterparts shows similar EERs on Vox1-O, E, and H when compared to ResNet34. Interestingly, we find that reversible variants can be constructed based on *Basic* block for ResNet101 and 152. For example, RevNet126 and 137, both relying on *Basic* block, achieve similar performance to ResNet101 with *Bottleneck* block. This phenomenon highlights the superior modeling capacity of *Basic* block for reversible networks. Moreover, *Basic*-based RevNets have fewer parameters compared to *Bottleneck*-based ones.

Table 1: *Architectural details and EER results of ResNets and our proposed RevNets on the Voxceleb1 dataset. All memory usage and maximum batch size are measured on a single 11GB 2080Ti GPU with 2-seconds utterances.*

System	Reversible	Depth	Width	Block	# Params	Memory (GB/utter)	Maximum Batch Size	Vox1-O	Vox1-E	Vox1-H
ResNet34	–	34	[32, 64, 128, 256]	Basic	6.6M	0.06	154	0.96	1.01	1.86
RevNet46	Partially	46	[48, 96, 192, 300]	Basic	6.7M	0.04	235	0.85	1.01	1.85
RevNet57	Fully	57	[48, 96, 192, 300]	Basic	6.1M	0.03	300	0.89	0.98	1.83
ResNet101	–	101	[32, 64, 128, 256]	Bottleneck	15.9M	0.33	31	0.62	0.80	1.48
RevNet126	Partially	126	[48, 96, 192, 384]	Basic	15.0M	0.04	213	0.58	0.80	1.48
RevNet137	Fully	137	[48, 96, 192, 384]	Basic	14.2M	0.03	297	0.58	0.80	1.49
RevNet140	Partially	140	[48, 96, 192, 300]	Bottleneck	15.8M	0.15	62	0.60	0.80	1.43
RevNet156	Fully	156	[48, 96, 192, 300]	Bottleneck	15.6M	0.12	73	0.59	0.81	1.45
ResNet152	–	152	[32, 64, 128, 256]	Bottleneck	19.8M	0.47	22	0.55	0.74	1.39
RevNet178	Partially	178	[48, 96, 192, 384]	Basic	18.3M	0.04	211	0.54	0.75	1.41
RevNet197	Fully	197	[48, 96, 192, 384]	Basic	18.2M	0.03	295	0.53	0.76	1.42
RevNet230	Partially	230	[48, 96, 192, 300]	Bottleneck	19.6M	0.20	48	0.49	0.72	1.33
RevNet246	Fully	246	[48, 96, 192, 300]	Bottleneck	19.4M	0.12	73	0.50	0.73	1.35

5.3. Memory Savings

To evaluate the memory consumption of each model, memory usage per utterance and maximum batch size are measured on a single 11 GB 2080Ti GPU using 2-seconds audio segments. From Table 1, we can see that the proposed reversible networks achieve notable memory savings under various scenarios. Specifically, for ResNet34, its reversible counterparts, namely RevNet46 and 57, demonstrate an approximate 2x memory savings. Moreover, both *Basic* and *Bottleneck*-based reversible variants of ResNet101 and 152 attain remarkable memory gains. By comparison, *Basic*-based RevNets exhibit more promising ability, achieving up to 15.7x reduction in memory usage compared to the vanilla networks. This can be attributed to the capability of reversible networks to reduce the necessity of storing intermediate activations during back-propagation. In addition, fully reversible networks has a greater advantage in memory savings over partial ones, which is reasonable since non-reversible downsampling layers in partially reversible networks are replaced with invertible operation.

5.4. Maximum Batch Size

For maximum batch size, we aim to determine the largest number of utterances that can be included in a batch without exceeding the GPU memory capacity. It displays an opposite trend to memory usage per utterance, as the preserved memory can be utilized to augment the training batch size. For example, RevNet57 can increase batch size by 1.9x compared to ResNet34. Similarly, RevNet197, the reversible variant of ResNet152, boosts the maximum batch size from 22 to 295 utterances. These encouraging results indicate the potential for training deeper models on consumer GPUs.

5.5. Analysis of Memory Usage and Parameter

In this section, we provide a comprehensive analysis of the correlation between the memory utilization and the parameter number for both ResNets and the proposed RevNets. As illustrated in Figure 1, ResNets exhibit a linear relationship between memory usage and network depth, which is consistent with the theoretical analysis provided in section 3.1. This is primarily due to the need to retain the intermediate activations during back-propagation, which scales up with the network depth. In contrast, the proposed RevNets demonstrate the ability to maintain consistent memory consumption even as network depth increases. Specifically, both Partially-RevNets (Basic) and Fully-RevNets (Basic) retain a fixed memory usage of 0.04GB and 0.03GB per utterance respectively, irrespective of network depth. This desirable property arises from the reversible nature of RevNets, which can perform back-propagation without the requirement to cache intermediate activations. As a result, memory consumption is decoupled from network depth. Moreover, fully reversible networks exhibit superior memory utilization efficiency when compared to partial ones. For example, Fully-RevNets (Bottleneck) achieve up to 40% memory reduction over Partially-RevNets (Bottleneck) with similar parameters and performance. In summary, the proposed RevNets can achieve significant memory savings of up to 15.7x with similar parameters and performance to the vanilla ResNets, indicating the great promise for memory-efficient training.

6. Conclusions

In this paper, we propose a novel family of Reversible Neural Networks (RevNets) for memory-efficient speaker verification. Specifically, two types of RevNets, namely partially and fully reversible networks, are introduced which reduce the necessity of storing intermediate activations during back-propagation. As a result, RevNets enjoy the advantage of retaining nearly constant memory usage as the depth of network increases. Experiments on Voxceleb demonstrate that compared to the vanilla ResNets, the proposed RevNets achieve up to **15.7x** memory savings with similar parameters and performance. To our knowledge, this is the first work to investigate memory-efficient training for speaker verification. Our results illustrate that reversible networks have the potential to serve as more efficient backbones for resource-constrained training scenarios.

7. Acknowledgement

This work was supported in part by China NSFC projects under Grants 62122050 and 62071288, and in part by Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102.

8. References

[1] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 531–542.

[3] D.Snyder, D.Garcia-Romero, G.Sell, D.Povey, and S.Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.

[4] H. Zeinali, S. Wang, A. Silnova, P. Matějka, and O. Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[5] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: emphasized channel attention, propagation and aggregation in tdnn based speaker verification," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2020, pp. 3830–3834.

[6] B. Liu, Z. Chen, and Y. Qian, "Attentive feature fusion for robust speaker verification," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2022, pp. 286–290.

[7] B. Liu, Z. Chen, S. Wang, H. Wang, B. Han, and Y. Qian, "Df-resnet: boosting speaker verification performance with depth-first design," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2022, pp. 296–300.

[8] Z. Chen, B. Liu, B. Han, L. Zhang, and Y. Qian, "The sjtu x-lance lab system for cnsrc 2022," *arXiv preprint arXiv:2206.11699*, 2022.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, and M. D. et al, "Tensorflow: A system for large-scale machine learning," in *Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Advances in Neural Information Processing Systems (NIPS) Autodiff Workshop*, 2017.

[12] R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," in *Machine Learning and Systems (MLSys)*, 2018.

[13] A. Gholami, Z. Yao, K. Sehoon, M. W. Mahoney, and K. Keutzer, "Ai and memory wall," *RiseLab Medium Post*, 2021.

[14] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4052–4056.

[15] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *INTERSPEECH*, 2017, pp. 999–1003.

[16] J. Thienpondt, B. Desplanques, and K. Demuynck, "The idlab voxsrc-20 submission: Large margin fine-tuning and quality-aware score calibration in dnn based speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5799–5803.

[17] M. Zhao, Y. Ma, M. Liu, and M. Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[18] P. Safari, M. India, and J. Hernando, "Self-attention encoding and pooling for speaker recognition," in *INTERSPEECH*, 2020, pp. 941–945.

[19] B. Han, Z. Chen, B. Liu, and Y. Qian, "Mlp-svnet : a multi-layer perceptrons based network for speaker verification," in *ICASSP*, 2022.

[20] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 2211–2221.

[21] R. Brügger, C. F. Baumgartner, and E. Konukoglu, "A partially reversible u-net for memory-efficient volumetric image segmentation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019, pp. 429–437.

[22] M. Pendse, V. Thangarasa, V. Chiley, R. Holmdahl, J. Hestness, and D. DeCoste, "Memory efficient 3d u-net with reversible mobile inverted bottlenecks for brain tumor segmentation," in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2020, pp. 388–397.

[23] Z. Cheng, B. Chen, G. Liu, H. Zhang, R. Lu, and Z. Wang, "Memory-efficient network for large-scale video compressive sensing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 246–16 255.

[24] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations (ICLR)*, 2019.

[25] Y. Zhao, S. Zhou, and Z. Zhang, "Multi-split reversible transformers can enhance neural machine translation," in *Association for Computational Linguistics (ACL)*, 2021, pp. 244–254.

[26] J. Huang and C. Wu, "Memory-efficient multi-step speech enhancement with neural ode," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2022, pp. 961–965.

[27] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[29] L. Dinh, J. Dickstein, and S. Bengio, "Density estimation using real nvp," in *International Conference on Learning Representations (ICLR)*, 2017.

[30] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2017, pp. 2616–2620.

[31] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2018, pp. 1086–1090.

[32] D. Snyder, G. Chen, and D. Povey, "Musan: a music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[33] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5220–5224.

[34] W. Cai, J. Chen, J. Zhang, and M. Li, "On-the-fly data loader and utterance-level aggregation for speaker and language recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 28, pp. 1038–1051, 2020.

[35] W. Wang, D. Cai, X. Qin, and M. Li, "The dku-dukeece systems for voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2010.12731*, 2020.

[36] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2019, pp. 2873–2877.

[37] Z. N. Karam, W. M. Campbell, and N. Dehak, "Towards reduced false-alarms using cohorts," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 4512–4515.