Supplementary Material: Reversible Vision Transformers

Karttikeya Mangalam* Haoqi Fan Yanghao Li
Chao-Yuan Wu Bo Xiong Christoph Feichtenhofer* Jitendra Malik

Appendix

This appendix contains further details for the main paper and is organized as follows: §A gives further architectural design details of both the Rev-ViT (Table A.1) and the Rev-MViT models (Table A.2). §B specifies training details including the pseudo-code for backpropagation (Algorithm 1) as well as hyperparameter settings for datasets across all tasks followed by a miscellaneous discussion of our work's potential impact on democratizing large model training (§C.1) & future work in speeding up large transformer training (§C.2) from a distributed systems view.

A. Architecture Details

Reversible Vision Transformers Table A.1 shows the architectures for all the Reversible Vision Transformer Models. All models closely follow the original ViT architectures [1] in matched performance, parameters, FLOPs and much lower memory footprint (Table 1). Output sizes denote the tensor shapes of the two residual streams at the end of each reversible Vision Transformer block. Note that even though the intermediate activations are twice the non-reversible variant, the actual memory needed is much lower because of memory reuse in reversible training. Further, the FLOPs are matched since each layer is performed only one of the two streams.

Reversible Multiscale Vision Transformers Table A.2 shows the architecture for the Rev-MViT-B model for image classification. The backbone is made-up of two stages – Stage-transition blocks that increase the channel capacity and down-sample the resolution and the reversible Stage-preserving blocks that form the majority of computation without changing feature dimensions. Similar to the Rev-ViT architecture, the output sizes o f both the streams are denoted. Fusion blocks operate on Y_1 and Y_2 together, hence operate with computationally light operations (Table 6).

B. Training Details

Reversible Back-propagation Pseudo-code. Algorithm 1 shows the peudocode for reversible backpropagation through the Rev-ViT-B block implemented in a similar fashion to Reversible ResNets [3]. The reversible transformation output O_1, O_2 is inverted to re-materialize the input activations I_1, I_2, which are then used to calculate the gradients of the parameters with respect to the loss function $\mathcal{L}$. Further, the gradients of the outputs $\nabla_{\mathcal{L}} O_1$ and $\nabla_{\mathcal{L}} O_2$ are propagated using chain rule and intermediate activations to calculate the gradients of the input $\nabla_{\mathcal{L}} I_1$ and $\nabla_{\mathcal{L}} I_2$.

While the pseudo-code routine delineates the procedure of Rev-ViT-B, a similar routine can handle stage preserving Rev-MViT Block simply by replacing the Multi-head attention (MHA) with the Multi-head pooling attention (MHPA) operation. Finally, since the stage transition Rev-MViT Block is non-reversible, automatic differentiation with activation caching is used to back-propagate through it.

stage	operators	output sizes
data		224×224
patch	1×16×16, 384 stride 1×16×16	384×14×14
rev	[**F** : MHA(384) **G** : MLP(1536)] ×12	[Y_1 : 384×14×14 Y_2 : 384×14×14]

(a) **Rev-ViT**-S with **4.6**G FLOPs, **22**M param, **8.8**MB/img memory, and **79.9**% top-1 accuracy.

stage	operators	output sizes
data		224×224
patch	1×16×16, 768 stride 1×16×16	768×14×14
rev	[**F** : MHA(768) **G** : MLP(3072)] ×12	[Y_1 : 768×14×14 Y_2 : 768×14×14]

(b) **Rev-ViT**-B with **17.6**G FLOPs, **87**M param, **17**MB/img memory, and **81.8**% top-1 accuracy.

stage	operators	output sizes
data		224×224
patch	1×16×16, 1024 stride 1×16×16	1024×14×14
rev	[**F** : MHA(1024) **G** : MLP(4096)] ×24	[Y_1 : 1024×14×14 Y_2 : 1024×14×14]

(c) **Rev-ViT**-L with **61.6**G FLOPs, **305**M param, **22.6**MB/img memory, and **81.4**% top-1 accuracy.

Table A.1. **Reversible Vision Transformer Architectures**: Rev-ViT are reversible adaption of ViT with exactly matched FLOPs, parameters and accuracy under identical conditions but with much lower GPU memory footprints.

stage	operators	output sizes
data		224×224
cubification	7×7, 96 stride 4×4	96×56×56
Stage-Preserving	$\begin{bmatrix} \mathbf{F}: \text{MHPA}(96) \\ \mathbf{G}: \text{MLP}(384) \end{bmatrix} \times 1$	$\begin{bmatrix} Y_1: 96\times56\times56 \\ Y_2: 96\times56\times56 \end{bmatrix}$
Stage-Transition	$\begin{bmatrix} \text{FUSION}(192) \\ \text{MHPA}(192) \\ \text{MLP}(768) \end{bmatrix} \times 1$	192×28×28
Stage-Preserving	$\begin{bmatrix} \mathbf{F}: \text{MHPA}(192) \\ \mathbf{G}: \text{MLP}(768) \end{bmatrix} \times 1$	$\begin{bmatrix} Y_1: 192\times28\times28 \\ Y_2: 192\times28\times28 \end{bmatrix}$
Stage-Transition	$\begin{bmatrix} \text{FUSION}(384) \\ \text{MHPA}(384) \\ \text{MLP}(1536) \end{bmatrix} \times 1$	384×14×14
Stage-Preserving	$\begin{bmatrix} \mathbf{F}: \text{MHPA}(384) \\ \mathbf{G}: \text{MLP}(1536) \end{bmatrix} \times 10$	$\begin{bmatrix} Y_1: 384\times14\times14 \\ Y_2: 384\times14\times14 \end{bmatrix}$
Stage-Transition	$\begin{bmatrix} \text{FUSION}(768) \\ \text{MHPA}(768) \\ \text{MLP}(3072) \end{bmatrix} \times 1$	768×7×7
Stage-Preserving	$\begin{bmatrix} \mathbf{F}: \text{MHPA}(768) \\ \mathbf{G}: \text{MLP}(3072) \end{bmatrix} \times 1$	$\begin{bmatrix} Y_1: 768\times7\times7 \\ Y_2: 768\times7\times7 \end{bmatrix}$

Table A.2. **Rev-MViT**-B with **8.7**G FLOPs, **39**M param, **66.8**MB/img memory, and **82.5**% top-1 accuracy is reversible adaptation of MViT-B architecture [1].

Training Hyperparameter	ViT-B	Rev-ViT-B
Learning Rate	1e-4	7e-5
Random augment Repeats (N)	1	2
Random augment Magnitude (M)	9	7
Optimizer Momentum	(0.9, 0.95)	(0.9, 0.999)
Weight Decay	0.3	0.3
Batch Size	4096	4096
Epochs	200	200
Label Smoothing	0.1	0.1
Drop Path Rate	0.2	0.2
Mixup	0.8	0.8
Cutmix	1.0	1.0

Table A.3. **Training Recipe for ViT-L and Rev-ViT-L**

ImageNet. Table A.3 shows the training recipes for ViT-L and Rev-ViT-L models presented in Table 1 of the main paper. Note that ViT-L is quite heavy with 61.6 GFLOPs and hence we adopt a shorter 200 epochs recipe for faster experiment cycle for developing Rev-ViT-L. Smaller ViT models – ViT-S and ViT-B – are trained according to the Data efficient transformers [6] and are all trained for 300 epochs. Hence, the accuracy difference between ViT-L which achieves 81.5% while ViT-B achieves 81.8% overall. MViT-B model follows the 300 epochs recipe as well proposed in [2].

Kinetics-400 & Kinetics-600. We follow the recipes proposed in [2] to train the Rev-MViT-B architecture (Table A.2) with crucial modifications in augmentations, as discussed in paper (Table 5).

MS-COCO. For object detection experiments, we adopt the Mask R-CNN [4] object detection framework in Detectron2 [7]. We follow the same training settings from [**?**], AdamW optimizer [5] ($\beta_1, \beta_2 = 0.9, 0.999$, base learning rate $1.6\text{e}{-4}$ for base size of 64, and weight decay of 0.1), and 3x schedule (36 epochs). The drop path rate is set as 0.4. We use PyTorch's automatic mixed precision during training.

All the proposed models, recipes and pretrained models for all the datasets will be made publicly available as well.

Algorithm 1 Backpropagation Through Rev-ViT Block

function REVVIT-BP($(O_1, O_2), (\nabla_{\mathcal{L}} O_1, \nabla_{\mathcal{L}} O_2)$)
  $z \leftarrow O_1$
  $I_2 \leftarrow O_2 - \text{MHA}(z)$
  $I_1 \leftarrow O_1 - \text{MLP}(I_2)$
  ▷ Automatic differentiation to calculate ∇_z & ∇_{I_2}.
  $\nabla_{\mathcal{L}} z \leftarrow \nabla_{\mathcal{L}} O_1 + \nabla_z \text{MHA}(z)^\top \nabla_{\mathcal{L}} y_2$
  $\nabla_{\mathcal{L}} I_2 \leftarrow \nabla_{\mathcal{L}} O_2 + \nabla_{I_2} \text{MLP}(I_2)^\top \nabla_{\mathcal{L}} z$
  $\nabla_{\mathcal{L}} I_1 \leftarrow \nabla_{\mathcal{L}} z$
  ▷ Automatic differentiation for $\nabla_{w_{\text{MLP}}}$ & $\nabla_{w_{\text{MHA}}}$.
  $\nabla_{\mathcal{L}} w_{\text{MLP}} \leftarrow \nabla_{w_{\text{MLP}}} \text{MLP}(I_2)^\top \nabla_{\mathcal{L}} z_1$
  $\nabla_{\mathcal{L}} w_{\text{MHA}} \leftarrow \nabla_{w_{\text{MHA}}} \text{MHA}(z)^\top \nabla_{\mathcal{L}} O_2$
  return (I_1, I_2) and $(\nabla_{\mathcal{L}} I_1, \nabla_{\mathcal{L}} I_2)$ and $(\nabla_{\mathcal{L}} w_{\text{MHA}}, \nabla_{\mathcal{L}} w_{\text{MLP}})$
end function

C. Discussion

C.1. Democratizing Large Model Training

Visual recognition backbones support a great variety of machine learning workloads in various forms and walks of life. Yet, the current research on large models remains constrained by the necessity of access to expensive GPU clusters that are required to train these models. Most research groups, including a large part of academia, cannot afford to develop these models because of these towering compute requirements. Efforts like ours, on memory-efficient high performing visual recognition backbone training democratizes the training of large models to several more groups with more limited compute budgets. For example, with a 15.5x per-example training memory reduction, Rev-ViT-L can be trained on a **single 8-GPU machine** with the training recipe (with the same batch size) that is developed with **16 8-GPU machines (128 GPUs)** for the ViT-L model. We hope that this democratization of large-scale allows other research groups to *meaningfully* contribute to this exciting research direction and decentralizes large-scale model development.

C.2. Faster Model Training

Reversible Vision Transformers change the order of computation in the backpropagation step of network training. Recomputation in backward order substitutes the intermediate

activation caching in forward pass, which allows an overall memory footprint reduction during training. Naturally, this activation rematerialization introduces some additional computational load in the backpropagation step. However, this is comfortably compensated for *deep models that yield higher training throughput* despite the computational burden because of the furnished lighter memory footprint.

Further, the additional back-propagation computational burden can also be overcome for smaller models. Transformer training is in a memory-bound regime, a reversible network specific distributed training schedule can significantly lighten or even completely hide the additional back-propagation latency behind the memory-bandwidth bottleneck (*i.e*. time spent in moving data from DRAM GPU memory to the SRAM and compute).

The sequential nature of first recomputing activations and then using them for gradient calculation can be substituted for a well instrument low-level kernel that can compute the activations in sync with the gradients in the backward pass. Conceptually, this would be much faster than the sequential implementation while analytically resulting in identical gradients. This requires a different focus from that of this work but can potentially speed-up the training throughput multiple times for reversible architectures and would make for excellent future work.

Acknowledgements. The authors would like to thank Harshayu Girase for help with benchmarking models; Amir Gholami, Ajay Jain and Nikita Kiatev for helpful research discussions and reference suggestions; Assaf Shocher, Matthew Tancik and Hang Gao for writing discussions and Shubh Gupta, Suzie Petryk, Hang Gao, Abhinav Agarwal, Medhini Narasimhan and Amur Ghosh for proofreading the manuscript and debugging typos; and BAIR industry collboration programs for providing computing resources.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. ICLR*, 2021. 1, 2

[2] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proc. ICCV*, 2021. 2

[3] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2211–2221, 2017. 1

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. ICCV*, 2017. 2

[5] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 2

[6] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In *icml*, 2021. 2

[7] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2