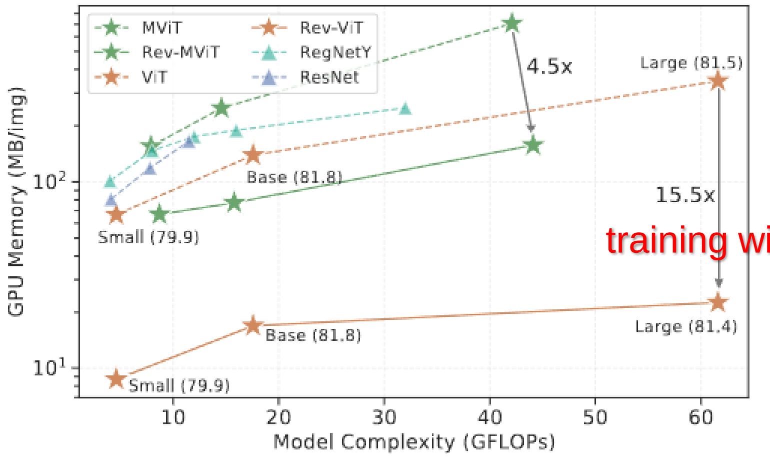


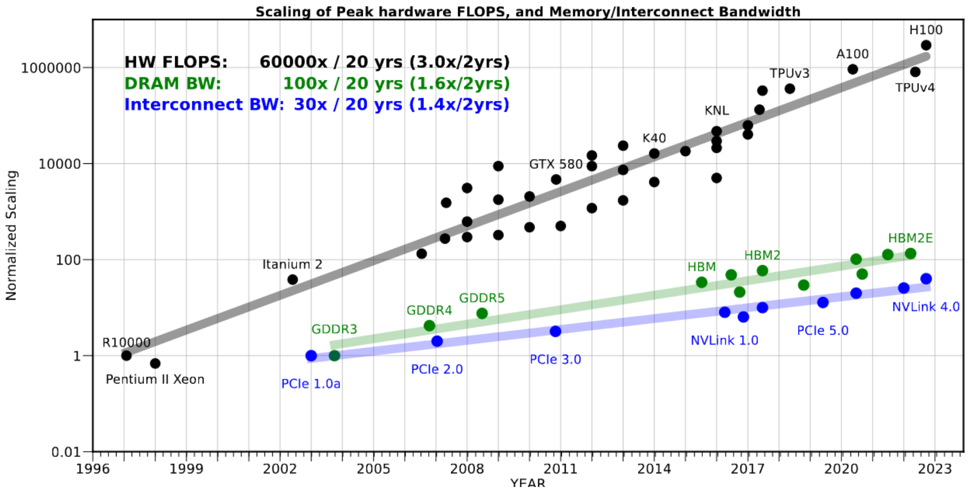
training with 1/15 memory

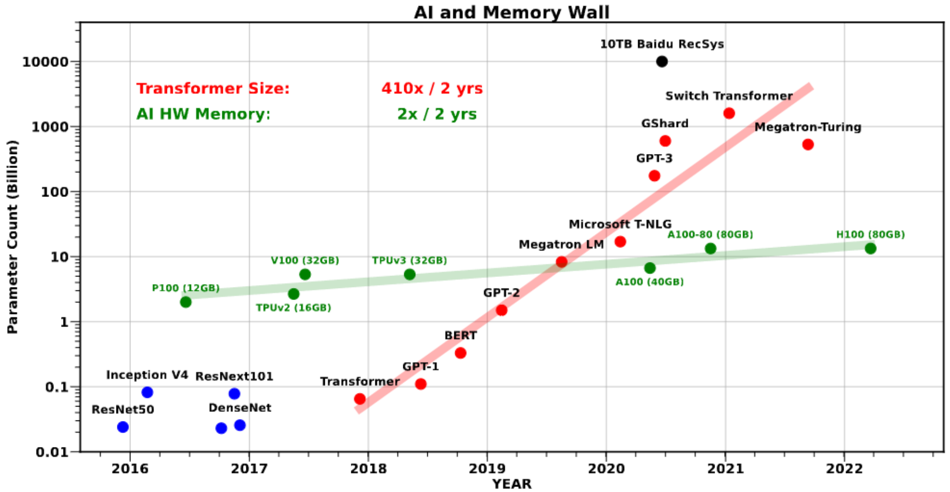
Reversible Vision Transformers
(CVPR 2022 Oral)

2024-11-29

손형욱 hyounguk.shon@kaist.ac.kr

Why memory-efficient training?



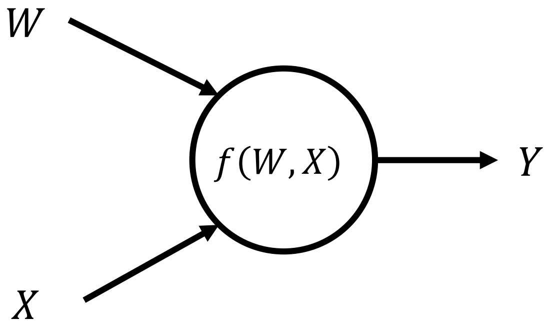


- Recent trends in GPU compute vs. memory
  - FLOPS has been increasing at a rate of ~3x every 2 years
  - Memory bandwidth only scales at a rate of ~1.6x every 2 years
  - Memory capacity only scales at a rate of ~2x every 2 years
  - **“Memory wall” = Most AI training & inferencing workloads are memory-bound.**

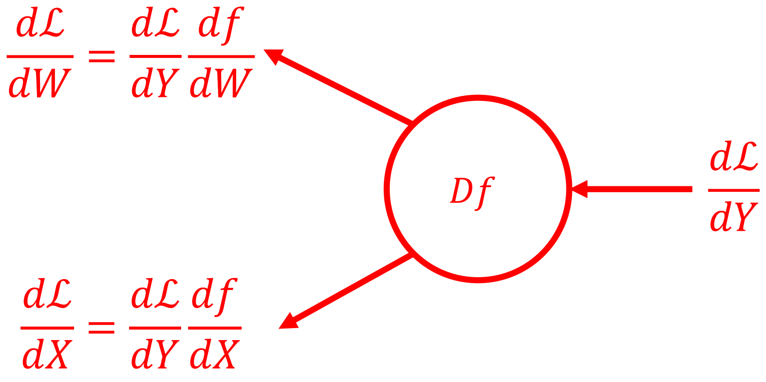
Gholami, Amir, et al. "AI and memory wall." *IEEE Micro* (2024).

Why memory-efficient training?

Forward pass



Backward pass



Consider the back-propagation mechanism. Given a computation graph node, $\mathcal{M}$, its children nodes $\{\mathcal{N}_j\}$, and the gradients of the children node with respect to final loss $\left\{\frac{d\mathcal{L}}{d\mathcal{N}_j}\right\}$, the back-propagation algorithm uses the chain rule to calculate the gradient with respect to $\mathcal{M}$ as,

$$\frac{d\mathcal{L}}{d\mathcal{M}} = \sum_{\mathcal{N}_j} \left(\frac{\partial f_j}{\partial \mathcal{M}}\right)^T \frac{d\mathcal{L}}{d\mathcal{N}_j}$$

$$\frac{d\mathcal{L}}{dW} = \left(\frac{d\mathcal{L}}{dY}\right) X^T \qquad \frac{d\mathcal{L}}{dX} = W \frac{d\mathcal{L}}{dY}$$

- Backpropagation has a high memory footprint
  - Activation X is needed compute the parameter gradient $d\mathcal{L}/dW$.
  - Requires caching intermediate activations computed through the forward pass
  - Peak memory footprint ∝ network depth D

Reducing the memory cost

Things that occupy the GPU memory:

- Activations saved for backward pass -- $|a|$
- Model parameters -- $|\Theta|$
- Optimizer state -- $2|\Theta|$ (for m_t and v_t in Adam)

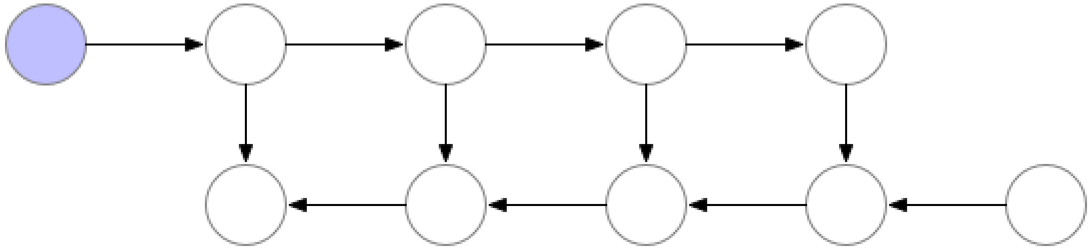
Language models: $|a| < |\Theta|$

Most other domains: $|a| \gg |\Theta|$

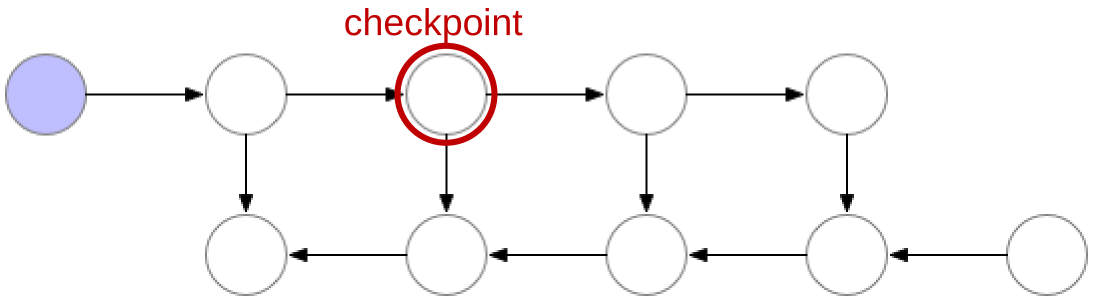
메모리 비율 그림

- Reducing the bit-precision of tensor dtype
  - bfloat16 format, Mixed precision training, Quantized training (QLoRA, 1-bit LLMs)
- Attaching trainable lightweight modules
  - Adapters, Low-rank adaptation (LoRA), Side-tuning
- Memory offloading / Model sharding
  - Offloading: Move large GPU tensors to CPU & SSD disk
  - Sharding: Partition a model across multiple GPUs
  - Microsoft DeepSpeed, PyTorch FSDP

Reducing the memory cost



Vanilla backprop



Grad checkpointing + backprop

- Trading compute for less memory appears to be a viable strategy
- Activation re-computation
  - Gradient checkpointing, FlashAttention variants
  - Implicit models (ODE network, fixed-point network, …)
  - Invertible models

Invertible models

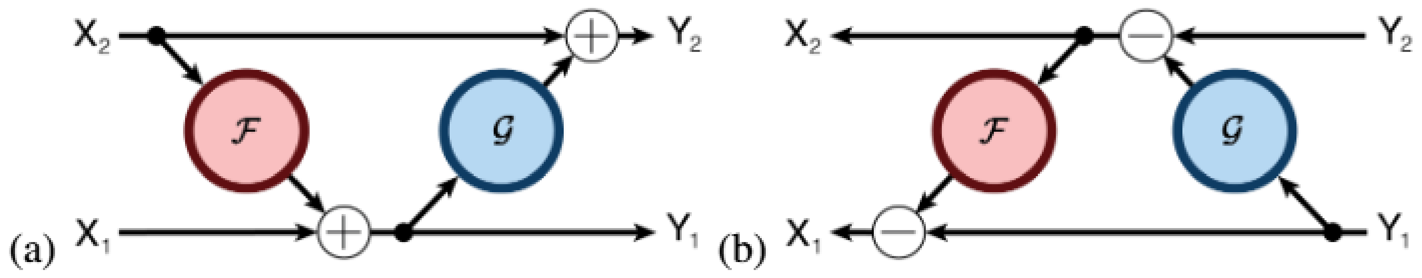


Figure 2: (**a**) the forward, and (**b**) the reverse computations of a residual block, as in Equation 8.

- Invertible Neural Networks
  - The activation dimension is partitioned into equal-sized vectors $\mathrm{X} = [\mathrm{X}_1 \quad \mathrm{X}_2]$.
  - $F(\cdot)$ and $G(\cdot)$ are typical non-reversible FFN such as MLP
  - The model is invertible $x = T^{-1}(y)$ thanks to a clever skip-connection design.
  - Input_dim and ouput_dim must match to have reversibility (1-to-1 mapping)
  - Prior applications in generative models – “Normalizing flows” [NICE, RealNVP]
  - Backprop without storing activations by reversing the output during backward [RevNets]

[NICE] Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).
[RealNVP] Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP." *International Conference on Learning Representations*. 2022.
[RevNets] Gomez, Aidan N., et al. "The reversible residual network: Backpropagation without storing activations." *Advances in neural information processing systems* 30 (2017).

Reversible Vision Transformers

- Proposed reversible & memory-efficient ViT architectures (Rev-ViT, Rev-MViT)
- Training recipes to match the performance of the vanilla counterpart
- 8.2x ~ 15.5x lighter memory cost at training
- Benchmarked on image classification, action recognition, object detection.
- Deep Rev-ViT can achieve up to 2-4x throughput compared to vanilla ViT

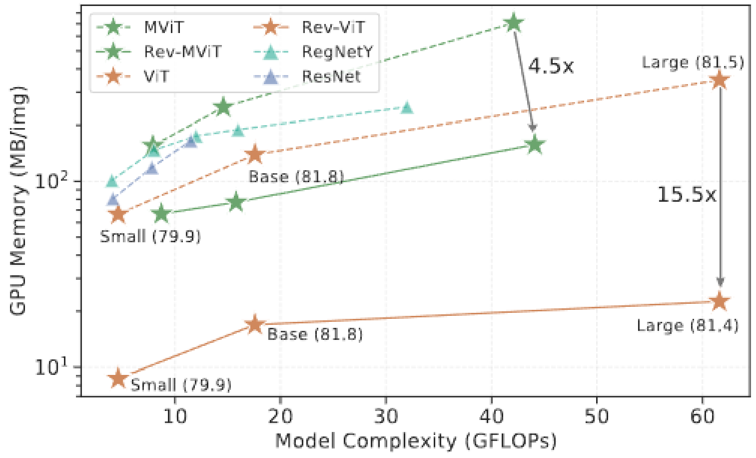
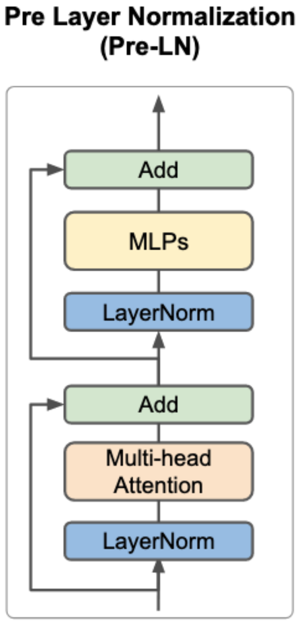
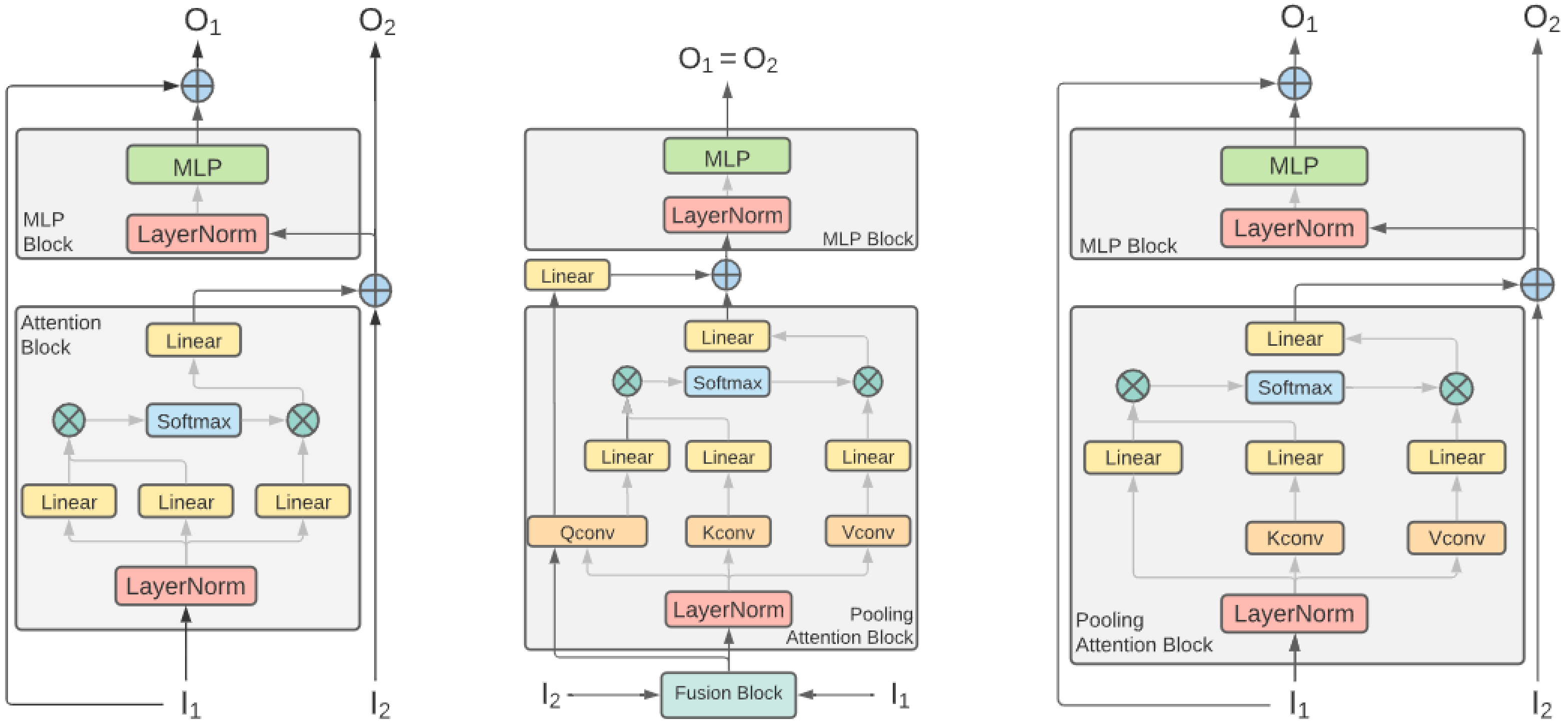


Figure 1. **Reversible Vision Transformers** are more memory-efficient, yet powerful *reversible counterparts* of state-of-the-art Vision Transformer (ViT) [15] and Multiscale Vision Transformer (MViT) [18] architectures with varying model complexity. Numbers in parentheses denote top-1 ImageNet performance. ResNet [28] and RegNet [58] are only shown for reference. For detailed discussion please refer to §4.1.

Reversible transformer blocks



Vanilla transformer



(a) Rev-ViT Block (b) Stage-Transition Rev-MViT Block (c) Stage-Preserving Rev-MViT Block

- Rev-ViT block
  - Adapting ViT to Two-Residual-Streams: Set F=attention block, G=MLP block
  - Boundary conditions: set $I_1 = I_2$ as the initial input to the reversible layers
  - Reconfiguring Residual Connections: no internal skip connections – found detrimental in training

Reversible MViT

- Multiscale ViT architecture [MViT]
    - ViT with gradually decreasing spatial resolution
    - Feature hierarchy of multiple resolution & channel widths
- Reversible MViT
    - Stage Transition block
        - Fuse I_1 and I_2, apply spatial pooling & channel upsampling
        - Non-reversible, so the activations must be cached
        - This is ok because the model has only a few of this block
    - Stage-Preserving block
        - Does not change the feature shape for reversibility
        - Reversible, so you don't cache the activations
        - Constitues most of the model.
- The model does not have to be end-to-end reversible
    - you still save memory with a few non-reversible components

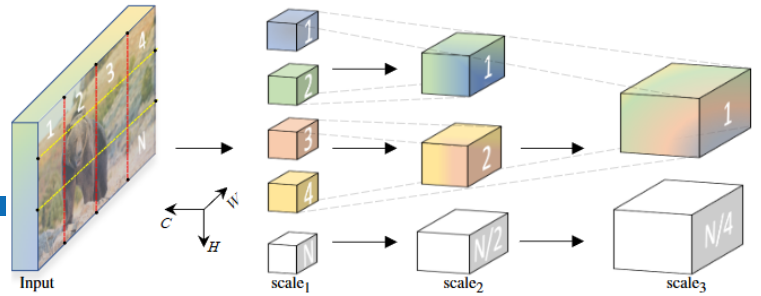
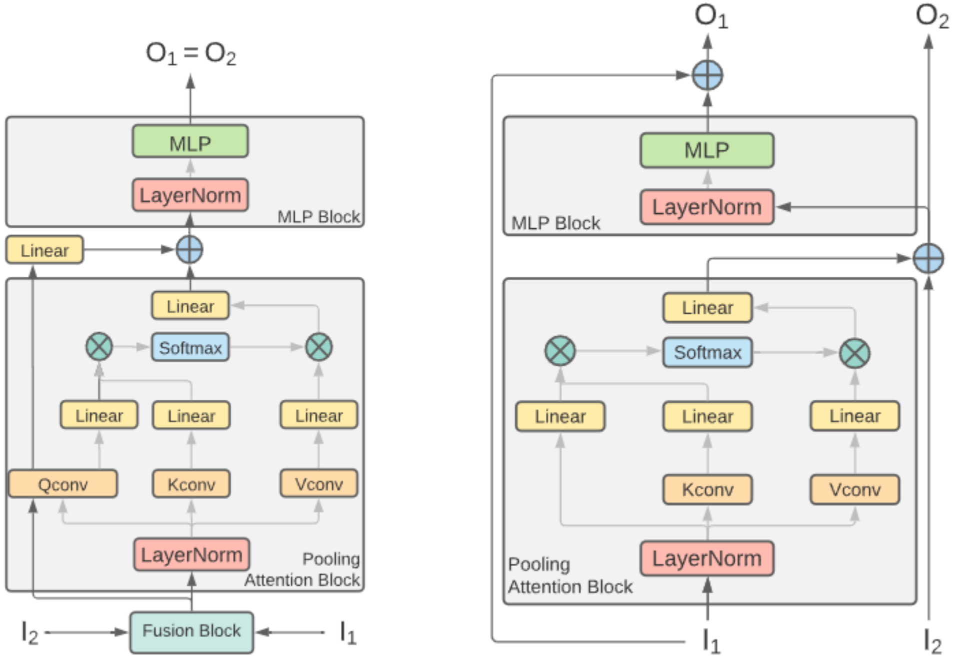


Figure 1. **Multiscale Vision Transformers** learn a hierarchy from *dense* (in space) and *simple* (in channels) to *coarse* and *complex* features. Several resolution-channel *scale* stages progressively *increase* the channel capacity of the intermediate latent sequence while *reducing* its length and thereby spatial resolution.



(b) Stage-Transition Rev-MViT Block (c) Stage-Preserving Rev-MViT Block

l-stream architecture composed of a stack of Reversible ViT blocks (a) that transforms the inputs n our reversible fashion. **Reversible MViT** is a two-residual-stream architecture as well, made The stage-transition blocks that act as coupling between the residual streams as well as perform sampling and (c) the stage-preserving blocks that form the majority of the computational graph nt feature dimension.

[MViT] Fan, Haoqi, et al. "Multiscale vision transformers." *Proceedings of the IEEE/CVF international conference on computer vision*. 2021.

Results

- Does Rev-ViT match the performance of vanilla ViT?
  - => **Yes, with significantly less memory footprint.**
- Benchmarks
  - Image classification (ImageNet dataset)
  - Video classification (Kinetics-400 & Kinetics-600)
  - Object detection (MS COCO)
- All results are trained from random initialization and a **single V100 16GB GPU**.

Results

- Image classification (ImageNet)
- Same FLOPs and parameter size with ViT
- Increasing memory savings with depth
  - 7.5x reduction (ViT-S) -> 15.5x reduction (ViT-L)
  - smaller memory cost means larger batch size
- Rev-MViT

model	Acc	Memory (MB/img)	Maxiumum Batch Size	GFLOPs	Param (M)
ResNet-101 [29]	76.4	118.7	112	7.6	45
ResNet-152 [29]	77.0	165.2	79	11.3	60
RegNetY-4GF [58]	80.0	101.1	136	4.0	21
RegNetY-12GF [58]	80.3	175.2	75	12.1	51.8
RegNetY-32GF [58]	80.9	250.2	46	32.3	32.3
Swin-T [48]	81.3	-	-	4.5	29
ViT-S [63]	79.9	66.5	207	4.6	22
Rev-ViT-S	79.9	**8.8** ↓7.5×	**1232** ↑5.9×	4.6	22
ViT-B [63]	81.8	129.7	95	17.6	87
Rev-ViT-B	81.8	**17.0** ↓7.6×	**602** ↑6.3×	17.6	87
RegNetY-8GF [58]	81.7	147.2	91	8.0	39
CSWin-T [14]	82.7	-	-	4.3	23
Swin-S [48]	83.0	-	-	8.7	50
ViT-L	81.5	349.3	26	61.6	305
Rev-ViT-L	81.4	**22.6** ↓15.5×	**341** ↑13.1×	61.6	305
MViT-B-16 [18]	82.8	153.6	89	7.8	37
Rev-MViT-B-16	82.5	**66.8** ↓2.3×	**157** ↑1.8×	8.7	39

Table 1. **Comparison to prior work on ImageNet-1K classification.** All memory and maximum batch size are on 224×224 input resolution on a 16G V100 GPU. **Rev-ViT** and **Rev-MViT** match performance across different FLOP regimes at a fraction of the per-input GPU memory cost.

Results

- Video classification (K-400, K-600 dataset)
- Matching performance with only 1/3 and ½ the memory cost

model	top-1	Mem (GB)	Max BS	GFLOPs×views	Param
Two-Stream I3D [5]	71.6	-	-	216 × NA	25.0
R(2+1)D [66]	72.0	-	-	152×115	63.6
Two-Stream R(2+1)D [66]	73.9	-	-	304 × 115	127.2
Oct-I3D + NL [8]	75.7	-	-	28.9×3×10	33.6
ip-CSN-152 [65]	77.8	-	-	109×3×10	32.8
SlowFast 4×16, R50 [19]	75.6	-	-	36.1 × 30	34.4
SlowFast 8×8, R101 [19]	77.9	-	-	106 × 30	53.7
SlowFast 8×8 +NL [19]	78.7	-	-	116×3×10	59.9
ViT-B-VTN-IN-1K [52]	75.6	-	-	4218×1×1	114.0
ViT-B-VTN-IN-21K [52]	78.6	-	-	4218×1×1	114.0
MViT-B-16 , 16×4	78.4	1.27	10	70.5×1×5	36.6
Rev-MViT-B-16, 16×4	78.5	**0.64**	**20**	64×1×5	34.9

Table 2. **Comparison to prior work on Kinetics-400 video classification.** Single view inference cost is reported along with used number of views (FLOPs×$\text{view}_{\text{space}}$×$\text{view}_{\text{time}}$). Memory (Mem) reported in Gigabytes per input clip. Maximum Batch Size (Max BS) measured as the maximum possible single GPU batch size. All measurements are performed on a single 16G V100 GPU.

model	top-1	Mem (GB)	Max BS	GFLOPs×views	Param
SlowFast 16×8 +NL [19]	81.8	-	-	234×3×10	59.9
X3D-XL	81.9	-	-	48.4×3×10	11.0
ViT-B-TimeSformer-IN-21K [3]	82.4	-	-	1703×3×1	121.4
ViT-L-ViViT-IN-21K [1]	83.0	-	-	3992×3×4	310.8
MViT-B-16, 16×4	81.3	-	-	70.3×1×5	36.6
MViT-B-16, 32×3	83.4	-	-	170×1×5	36.8
MViT-B-24, 32×3	83.8	4.40	2	236×1×5	52.9
Rev-MViT-B-24, 32×3	83.7	**1.64**	7	223×1×5	51.8

Table 3. **Comparison to prior work on Kinetics-600 video classification.** Results under same settings as Kinetics-400 in Table 2.

Results

- Object detection (MS COCO)
- ImageNet-pretrained weights
- RevViT feature backbone with Mask-RCNN detector
- Matching performance with ½ memory cost

Model	AP^{box}	AP^{mask}	Memory(GB)	GFLOPs	Param (M)
Res50 [28]	41.0	37.1	-	260	44
Res101 [28]	42.8	38.5	-	336	63
X101-64 [73]	44.4	39.7	-	493	101
PVT-L [69]	44.5	40.7	-	364	81
MViT-B	48.2	43.9	18.9	668	57
Rev-MViT-B	48.0	43.5	10.9	683	58

Table 4. **Comparison on MS-COCO object detection.** Rev-MViT achieves competitive performance to MViT across all metrics at **1.7**× lower memory footprint.

Results

- Training recipe
  - Rev-ViT seems to have inherent regularization
  - Lighter augmentation & higher weight decay
- Designing the lateral fusion layer
  - Ways to produce $I = \mathrm{fusion}(I_1, I_2)$.
  - Fusion layer in State-transition block
  - Fusion at the encoder feature output

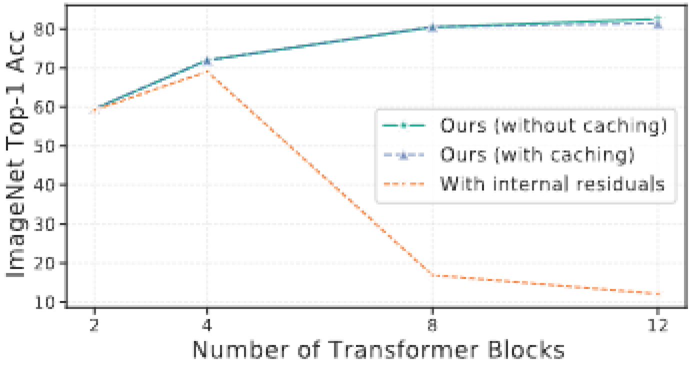
Training Improvement	Train Acc	Top-1 ImageNet Acc
Naïve Rev-ViT-B	15.3	12.1
+ Re-configuring residual streams	82.1	77.2
+ Repeated Augmentation	84.9	80.6
+ Lighter Augmentation magnitude	93.2	81.0
+ Stronger Stochastic Depth	92.0	81.4
+ Higher weight decay	91.0	81.8
Rev-ViT-B	91.0	81.8

Table 5. **Rev-ViT-B Training Recipe.** We observe that reversible transformers tend to have a stronger inherent regularization and require a lighter augmented training recipe for peak performance.

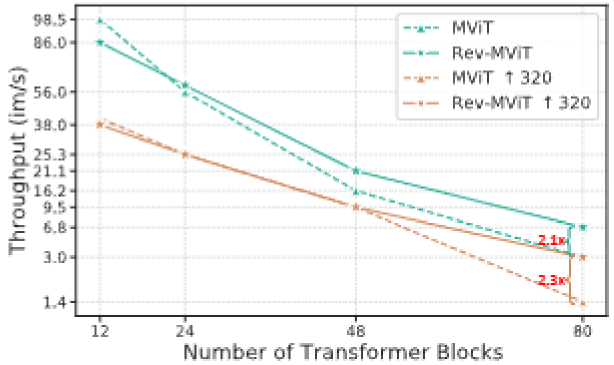
Stage-Transition Fusion	Termination Fusion	Train Acc	Top-1 Acc
Max	Norm $\rightarrow$ Concat	78.1	81.7
Concat	Norm $\rightarrow$ Concat	79.1	82.0
2×-MLP	Norm $\rightarrow$ 2×-MLP	80.2	81.8
2×-MLP + 0.2 dp	Norm $\rightarrow$ 2×-MLP $\rightarrow$ 0.5dp	77.1	81.2
2×-MLP	Norm $\rightarrow$ 1-layer	53.6	82.1
2×-MLP	Norm $\rightarrow$ 1-layer $\rightarrow$ 0.2dp	64.0	82.4
Norm $\rightarrow$ 2×-MLP	Norm $\rightarrow$ Concat	79.4	82.3
Norm $\rightarrow$ 2×-MLP	Norm $\rightarrow$ 1-layer $\rightarrow$ 0.2dp $\rightarrow$ Norm	78.3	82.3
4×-MLP	Norm $\rightarrow$ Concat	80.4	82.3
2×-MLP	Concat $\rightarrow$Norm	80.5	82.2
2×-MLP	Norm $\rightarrow$ Concat	80.1	82.5

Table 6. **Lateral Fusion Strategies**. Residual streams I_1 and I_2 are fused in state-transition blocks (§3.3.1) as well as on termination (§3.2.2) before the network head. We find fusion strategy to

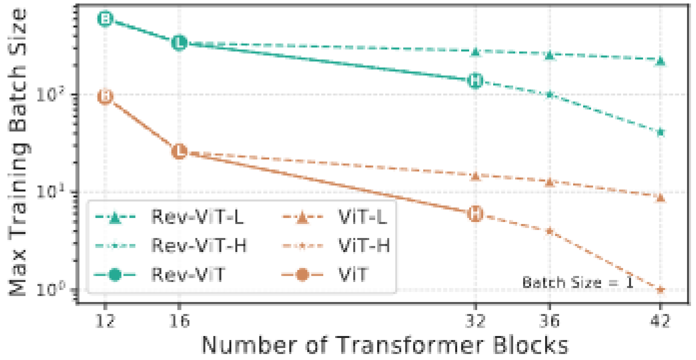
Results



(a) Activation caching and internal residuals.



(b) Training throughput vs. Model Depth



(c) Reversible training and maximum batch size.

- (a) Training stability – Activation inversion does not hurt accuracy.
- (b) Throughput -- Deep Rev-ViT have higher training throughput due to larger maximum batch size
- (c) Maximum batch size -- Rev-ViT allow much larger batch size vs. its counterparts

Following works

- Converting pre-trained transformers [Dr2Net, MEFT]
- Reversible Swin Transformer [Re2TAL]
- Efficient large-scale training [PETRA]
- Generalization to m-th order recursion [RevCol]
- Vertical integration of memory-efficient techniques

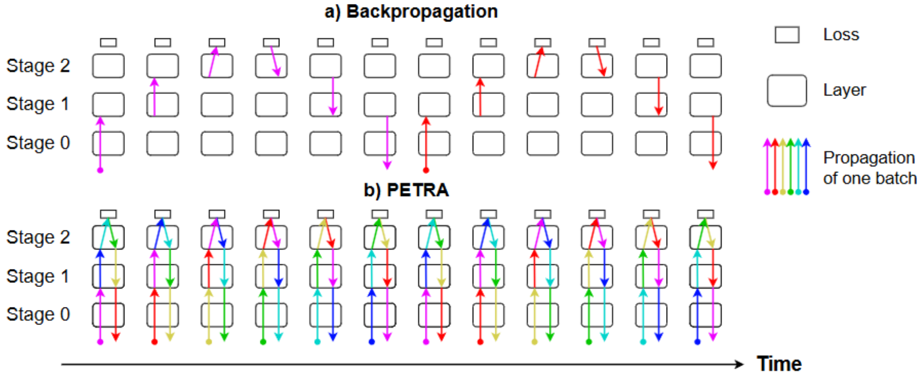


Figure 1: **Comparison of PETRA with standard backpropagation.** This approach splits the stages of a model and decouples their forward and backward passes, resulting in a sixfold increase in parallelization speed in this example.

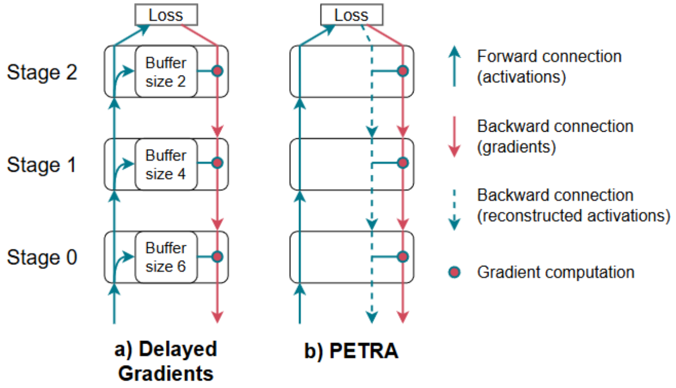


Figure 3: **Comparison of our PETRA method to a standard Delayed Gradient method [42].** By avoiding weight stashing and reversing the output into the input during the backward phase, we are able to fully decouple the forward and backward phases in all reversible stages, with no memory overhead, compared to standard delayed gradient approaches.

[Dr2Net] Zhao, Chen, et al. "Dr2Net: Dynamic Reversible Dual-Residual Networks for Memory-Efficient Finetuning." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.
[MEFT] Liao, Baohao, Shaomu Tan, and Christof Monz. "Make pre-trained model reversible: From parameter to memory efficient fine-tuning." *Advances in Neural Information Processing Systems* 36 (2024).
[Re2TAL] Zhao, Chen, et al. "Re2TAL: Rewiring pretrained video backbones for reversible temporal action localization." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.
[PETRA] Rivaud, Stéphane, et al. "PETRA: Parallel End-to-end Training with Reversible Architectures." *arXiv preprint arXiv:2406.02052* (2024).
[RevCol] Cai, Yuxuan, et al. "Reversible Column Networks." *The Eleventh International Conference on Learning Representations*.