Abstract

We present Reversible Vision Transformers, a memory-efficient architecture design for visual recognition. By decoupling the GPU memory footprint from the depth of the model, Reversible Vision Transformers enable memory-efficient scaling of transformer architectures. We adapt two popular models, namely Vision Transformer and Multiscale Vision Transformers, to reversible variants and benchmark extensively across both model sizes and tasks of image classification, object detection and video classification. Reversible Vision Transformers achieve a reduced memory footprint of up to $15.5 \times$ at identical model complexity, parameters and accuracy, demonstrating the promise of reversible vision transformers as an efficient backbone for resource limited training regimes. Finally, we find that the additional computational burden of recomputing activations is more than overcome for deeper models, where throughput can increase up to $3.9 \times$ over their non-reversible counterparts. Code and models are available at https://github.com/facebookresearch/mvit.

1. Introduction

The deep learning revolution in computer vision has rested on the bedrock of high performance hardware accelerators. Fueled by special purpose AI accelerators, the compute requirements for state-of-the-art models are growing exponentially. However, compute is only half the story. The other, and often overlooked half, is memory bandwidth bottleneck, which has been difficult to proportionally scale as compared to peak accelerator FLOPs [51]. In particular, the peak accelerator FLOPs have been increasing at a rate of $\sim 3.1 \times$ every 2 years. However, peak bandwidth only scales at a rate of $\sim 1.4 \times$ every 2 years. This disparity is exacerbated in transformers, which have been doubling in required compute roughly every three months for the past three years, resulting in a so-called memory wall [21] where both the overall model performance as well as the training speed have become tightly memory-bound [33].

As such, for bandwidth bound models, trading compute for memory through re-computation could actually be more efficient than using work-optimal algorithms [67, 68]. In the case of training neural network models, this can be achieved by re-computing activations instead of storing and then loading them from DRAM [30]. Besides training speed, scaling vision transformers in depth naturally hits the GPU memory capacity, especially in memory starved regimes such as video recognition where state-of-the-art models are often limited to batch size 1 due to high memory footprint of intermediate activations.

In this work, we propose Reversible Vision Transformers, a family of expressive visual recognition architectures with very favorable activation memory footprints (Figure 1) compared to their non-reversible variants. By trading-off GPU activation caching with efficient on-the-fly activation re-computation, reversible vision transformers effectively decouple the activation memory growth from the depth of the model. While the natural language processing community has performed some early exploration of reversible transformers for machine translation [37], these techniques focus on longer sequence lengths rather than depth.

Our experiments show that a straightforward adaptation of vision transformers to reversible architectures fails to scale for deeper models because of training convergence in-
stabilities due to internal sub-block residual connections.

In this work, we reconfigure the residual paths in Vision Transformers (ViT) [15] and Multiscale Vision Transformers (MViT) [18] to overcome this issue. We further find that reversible structures have stronger inherent regularization and therefore, we use a lighter augmentation recipe (repeated augmentation, augmentation magnitude and stochastic depth) and lateral connections between residual blocks.

We benchmark extensively across image recognition tasks such as image classification and object detection as well as video classification, across all of which, reversible vision transformers have competitive performance to their non-reversible counterparts suffering negligible to no performance decay. Moreover, reversible models have extremely favorable per-image memory footprint, saving 15.5× on the ViT-Large model and 2.3× on the MViT-Base model with reversible training.

In summary, our contributions are three-fold.

(i) We propose Reversible Vision Transformer (Rev-ViT) and Reversible Multiscale Vision Transformers (Rev-MViT), memory efficient reversible adaptations of state-of-the-art visual recognition backbones.

(ii) We observe reversible transformers to have a stronger inherent regularization than vanilla networks. Hence, we develop new training recipes by adapting the original recipes with different repeated augmentations, augmentation magnitudes and drop path rate to match the performance of their non-reversible counterparts.

(iii) We benchmark our models across several tasks: image classification, object detection and action recognition, across accuracy, memory, maximum training batch size and model complexity. In particular, at matched complexity (FLOPs/parameters) and final accuracy specifications, Rev-ViT-B and Rev-ViT-L train with per image memory footprints that are 7.6× and 15.5× lighter than ViT-B and ViT-L respectively. Further, we show how deep reversible networks can achieve up to 3.9× higher throughput than their non-reversible counterparts.

2. Related Work

Transformers are a popular network structure that were first proposed for natural language applications [65] and now are widely used in all areas of deep learning such as Reinforcement Learning [7], Speech [40], Music [31], multi-modal learning [34] and recently, in traditional vision tasks [15] as well. Since their introduction, Vision Transformers have experienced enthusiastic adoption and have been applied to several visual recognition tasks [15, 60, 61] using priors such as multi-scale feature hierarchies [18, 23, 47, 66, 72] and local structure modelling [9, 14, 47]. Further, vision transformers have also been generalized for action recognition and detection in videos [1, 3, 18, 47, 49, 50].

However, a crucial problem with scaling up transformer models is the growth of required GPU memory with depth. This linear growth in memory is prohibitive to the development of very deep models since the batch size needs to be reduced considerably to be able to accommodate storing the intermediate activations on GPU. This problem is exacerbated in video models which process very large input tensors and are often trained with batch size 1 even for shallower depths. A potential systems-level solution to scale up conventional transformer architectures is model parallelism [10] that puts different parts of the model on different GPUs. However in practice, it is quite slow and requires special high bandwith network infrastructure because of huge across device traffic.

In this work, we use Vision Transformers [15] and Multiscale Vision Transformers [18] as our base models and propose their reversible transformer version that decouple the memory requirement from depth of the model. This facilitates saving GPU memory and allows training with much higher batch size, and consequently, to preserve or even increase training throughput of deep non-reversible models.

Reversible Architectures are a family of neural network architectures that are based on the NICE [12, 13] reversible transformation model which are the precursors of the modern day generative flow based image generation architectures [29, 36]. Based on the NICE invertible transformations, Gomez et al. [22] propose a Reversible ResNet architecture that employs the reversible transformation [12] for memory-efficient image classification in ResNets [26]. An interesting line of work builds upon the Reversible ResNets ideas proposing better reversible CNN models using ODE characterizations [6, 38, 56], momentum [38, 56], layer-wise inversion [24], fourier transform based inversion [20] and fixed point iteration based inversion [2, 57]. Reversible CNNs have been applied to several traditional image tasks such as compression [45], reconstruction [42], retrieval [41], and denoising [32, 46] as well as to compressed sensing [58], compact resolution [71], image to image translation [64], remote sensing [53], medical image segmentation [52, 70] and MRI reconstruction [54]. Reversible transformation have also been adapted to other networks such as RNNs [48], Unet [4, 16], Masked Convolutional Networks [57] and 1000-layer deep Graph Neural Networks [39]. Some early attempts have also been made to adapt the reversible transformation to the NLP domain, initiated by Kiatev et al. [37] and built upon in [73, 74] for machine translation.

However, word-level input partitioning contains much richer semantic content than patch level image partitioning and NLP transformers tend to be focusing on longer sequences with a wider but shallower network. For example, Kiatev et al. [37] focus on expanding on the input sequence dimension rather than depth and without benchmarking on
max batch-size, peak GPU memory & training throughput.

Our experiments show that a naive adaption of the NICE reversible transformation to transformers, trained with the same recipe (augmentations, regularization etc.) as ViT-B, performs poorly for deeper (≥8 blocks) models. This work is the first to propose Reversible Vision Transformers, adapt it to two state-of-the-art transformers, namely, ViT & MViT and apply the reversible backbone for object detection and video classification, which tends to be one the most memory starved domains of visual recognition.

3. Approach

We first present a brief overview of the reversible transformation (§3.1.1) and its benefits in neural network training (§3.1.3). We then present our proposed Reversible Vision Transformer (§3.2) its two residual stream structure (§3.1.2) and associated constraints (§3.1.3). This is followed by our proposed Reversible Multiscale Vision Transformer (§3.2.1) and its sub-blocks (§3.3.2 and §3.3.1) that allow end-to-end reversible training.

3.1. Reversible Block Structure

The reversible transformer is composed of a stack of reversible blocks that follow the structure of the reversible transformation (§3.1.1) and its benefits in neural network training (§3.1.3). We then present our proposed Reversible Vi

3.1. Reversible Transformation

Consider a transformation $T_1$ that transforms an input tensor $I$ partitioned into two d dimensional tensors, $[I_1; I_2]$ into the output tensor $O$ also similarly partitioned into tensors, $[O_1; O_2]$ with an arbitrary differentiable function $F(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ as follows:

$$I = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \xrightarrow{T_1} \begin{bmatrix} O_1 \\ O_2 \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 + F(I_1) \end{bmatrix} = O$$

Note that the above transformation $T_1$ allows an inverse transformation $T_1'$ such that $T_1' \circ T_1$ is an identity transform. Also, consider an analogous transposed transformation $T_2$ using the function $G(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ as follows:

$$I = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \xrightarrow{T_2} \begin{bmatrix} O_1 \\ O_2 \end{bmatrix} = \begin{bmatrix} I_1 + G(I_2) \\ I_2 \end{bmatrix} = O$$

Similar to $T_1$, $T_2$ also allows an inverse transform $T_2'$. Now consider the composition $T = T_2 \circ T_1$ that transforms both the partitions of the input vector $I$ and is obtained as,

$$I = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \xrightarrow{T} \begin{bmatrix} O_1 \\ O_2 \end{bmatrix} = \begin{bmatrix} I_1 + G(I_2 + F(I_1)) \\ I_2 + F(I_1) \end{bmatrix} = O \quad (1)$$

Naturally, $T$ affords the inverse transform $T' = T_1' \circ T_2'$ that follows $T'(T(I)) = I$. Note that the inverse transform $T'$ queries $F$ and $G$ exactly once without explicitly inverting them and hence has the same computational cost as the forward transform $T$.

3.1.2 Vanilla networks require caching activations

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

$$\frac{dL}{dM} = \sum_j \left( \frac{\partial f_j}{\partial M} \right)^T \frac{dL}{dN_j}$$

where $f_j$ denotes the function computing node $N_j$ from its parents, $M$ being one of them. The jacobian $\frac{\partial f_j}{\partial M}$, requires calculating the partial gradient of the $f_j$ output with respect to the current node $M$.

Now consider the simplest possible neural network layer $f(X) = W^T X$, where $X$ is an intermediate activation inside the network. Applying the above described backpropagation algorithm to compute the derivative with respect to parent nodes, and using the output $Y$ as the sole child node, $N_j$, we get,

$$\frac{dL}{dW} = (\frac{dL}{dY}) X^T \quad \frac{dL}{dX} = W \frac{dL}{dY}$$

Thus, because of the function jacobian, the backpropagation algorithm requires intermediate activations during the forward pass to be available in the backward pass to compute the gradients with respect to the weights.

Typically, this is achieved by caching the intermediate activations on GPU memory for use in the backward pass. This allows fast gradient computation at the cost of extra memory. Further, the sequential nature of the network requires the activations for all the layers to be cached in before the loss gradients are calculated and the cached memory is freed. This dependence significantly affects the peak memory usage which thus becomes linearly dependent on the network depth $D$.

3.1.3 Learning without caching activations

As noted in §3.1.1, an input transformed with the reversible transformation $T$ allows recalculating the input from the output of the transformation. Hence, a network composed of such reversible transformations does not need to store intermediate activations since they can be recomputed easily in the backward pass from the output. However the reversible transformation $T$ places an important constraint on the property of the learnt function.

**Equidimensional Constraint.** As mentioned in §3.1.1, the functions $F$ and $G$ need to be equidimensional in input and output spaces to be reversible. Hence, the feature dimensions need to remain constant under $T$. While this constraint is an obstruction for other vision architectures
such as ResNets [26] that require a change of feature dimensions, it is easily satisfied in the Vision Transformer architecture [15] which maintains a constant feature stack throughout the layers.

3.2. Reversible Vision Transformers

3.2.1 Adapting ViT to Two-Residual-Streams

Fig. 2a shows the reversible transformation $T$ adapted to the Vision Transformer architecture [15]. The input consists of two partitioned tensors $I_1$ and $I_2$ that are transformed as per the equation 3.1.1 maintaining reversibility. This leads to a two-residual-stream architecture where each of the inputs $I_1$ and $I_2$ maintain their own residual streams while mixing information with each other using functions $F$ and $G$. Following ViT [15], we use the Multi-head attention as $F$ and the MLP subblocks as $G$ in the reversible transformation.

3.2.2 Boundary Conditions

As the ViT architecture only uses a single residual stream, the architecture needs to be modified to support the two-residual-stream design (§3.2.1). We propose the following: 1. Initiation. We keep the patchification stem intact and simply initialize both $I_1$ and $I_2$ identically as the patchification output activations. Note that this design choice is different from [22] which proposes to split in halves along the channel dimensions.

2. Termination. The two residual paths need to be fused before the final classifier head to preserve information. We propose to layer-normalize the inputs first, followed by concatenation, to reduce the fusion computational overhead.

3.2.3 Reconfiguring Residual Connections

Residual connections play a key role for signal propagation in deep networks [26]. The reversible transform $T$ itself also depends crucially on the residual connections between the two streams to maintain reversibility. Interestingly, we observe a key relationship between the residual connections and signal propagation in Reversible Vision Transformer.

Note that while it is common practice for neural network blocks to be wrapped around a residual connection for better gradient flow [26], there is no such connection for either the $I_1$ or $I_2$ inputs. Specifically, internal residual connections around the MLP and attention sub-blocks for both the $I_1$ and $I_2$ streams are absent. Instead, the residual connections for each residual stream flows through the other stream, operating through the inherent skip connection present in the reversible transformation $T$ (§3.1.1). We find these internal skip connections detrimental to training convergence for deeper models while bringing no additional gain for shallower models and choose to omit them entirely for reversible vision transformer blocks.

Figure 2. Reversible ViT is a two-residual-stream architecture composed of a stack of Reversible ViT blocks (a) that transforms the inputs $I_1$ and $I_2$ with the ViT design [15], but in our reversible fashion. Reversible MVIT is a two-residual-stream architecture as well, made up of a stack of two type of blocks – (b) The stage-transition blocks that act as coupling between the residual streams as well as perform channel upsampling and resolution downsampling and (c) the stage-preserving blocks that form the majority of the computational graph and propagate information preserving input feature dimension.
3.3. Reversible Multiscale Vision Transformers

The recently proposed MViT architecture develops a feature hierarchy inside the model by downsampling the visual resolution and upsampling the channel dimension. It obtains state-of-the-art results on both image and video classification benchmarks. To showcase the flexibility of the reversible design, we adapt the MViT model to Reversible Multiscale Vision Transformers. We propose to compose the Reversible MViT architecture in the same structure as the MViT model but using two different layers – the Stage Transition and the Stage-Preserving blocks.

3.3.1 Stage-Transition Block

Figure 2b depicts the architecture of the proposed stage-transition block. The stage-transition block closely follows the design of the resolution upsampling blocks in MViT [18] with the following crucial modifications:

Lateral Connections. The residual streams $I_1$ and $I_2$ are fused with lateral connections at the start of the stage-transition block. This allows efficient computation of the resolution downsampling and feature upsampling without repeat computation in each stream separately.

Feature Upsampling. MViT performs feature upsampling in the last MLP block before the resolution upsampling block. We propose to move the channel upsampling stage inside the pooling attention sub-block of the stage-transition block. Specifically, we propose to upsample the Query, Key and Value vectors in the linear layer following the pooling channel-wise convolutional layers (Figure 2b and 2c). This was the dual benefit of (A) allowing all feature dimension changes to take place in sync inside the same block and allowing other blocks to keep feature dimensions intact, a virtue of reversible architectures ($\S$3.1.3) and (B) saving additional computation from being used in the prior MLP and pooling layers.

We follow the same boundary conditions at the stage-transition blocks as in the reversible vision transformer architecture ($\S$3.2.2).

3.3.2 Stage-Preserving Block

Figure 2c shows the reversible transformation $T$ ($\S$3.1.1) adapted to the Multiscale Vision Transformer architecture [18]. The design closely resembles that of the reversible vision transformer block (Figure 2a) with the addition of multi-head pooling attention [18]. Note that even though the attention uses pooling on key and value tensors, thereby changing the sequence length, the output dimensions are still preserved. Hence, the stage-preserving block still follows the equidimensional constraint ($\S$3.1.3) and hence can be made fully reversible and learnt without caching activations.

Since each stage-transition block changes the spatiotemporal resolution, they occur only a limited number of times in the entire MViT network. In other words, the majority of the computation as well as memory usage is performed within the stage-preserving blocks and is fully reversible. We follow the same residual connection circuit ($\S$3.2.3) as in Reversible Vision Transformer blocks for both the stage-transition and the stage-preserving blocks. Further details of the backpropagation procedure for the Rev-ViT and both the Rev-MViT blocks are presented in the supplementary.

4. Results

Datasets. We benchmark both the Reversible Vision Transformer and the Reversible Multiscale Vision Transformer architectures extensively across image classification (ImageNet [11]), video classification (Kinetics 400 [35] & Kinetics 600 [5]) and object detection (MS-COCO [44]). Across all the benchmarks, we observe significant memory savings by using the reversible architecture with negligible to no accuracy change. All presented results and ablations are trained from random initialization, except for object detection on COCO where we initialize from the corresponding ImageNet weights.

4.1. Image Classification

Settings. We benchmark our proposed models on image classification on the ImageNet-1K dataset [11] with $\sim$1.28M images among 1000 classes. We follow training recipes [17] for both ViT [15] and MViT [18] models with certain crucial adaptions ($\S$6). All models are trained from random initialization without EMA for 300 epochs except for ViT-L and Rev-ViT-L which follow a 200 epoch training recipe. Training details are in the appendix.

Results. Table 1 shows the results for Reversible Vision and Reversible Multiscale Vision Transformers across different models and FLOP regimes. We benchmark the memory and maximum batch sizes of all the models on a single 16 GB V100 GPU under $224 \times 224$ image size and otherwise identical conditions. The maximum batch size is obtained as the highest number of images in a batch than can train without running out of GPU memory. The memory per image is measured as the peak GPU memory each image occupies during training.

Specification Match. We note that Reversible Vision Transformers match the FLOP, parameter and inference memory of their non-reversible counterparts exactly, owing to the parsimonious design of the Rev-ViT block ($\S$3.2). The Reversible Multiscale Vision Transformer can have
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.

<table>
<thead>
<tr>
<th>model</th>
<th>Acc (16G V100)</th>
<th>Memory (MB/img)</th>
<th>Maximum Batch Size</th>
<th>GFLOPs</th>
<th>Param (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-101</td>
<td>76.4</td>
<td>118.7</td>
<td>112</td>
<td>7.6</td>
<td>45</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>77.0</td>
<td>165.2</td>
<td>79</td>
<td>11.3</td>
<td>60</td>
</tr>
<tr>
<td>RegNetY-4GF</td>
<td>80.0</td>
<td>101.1</td>
<td>136</td>
<td>4.0</td>
<td>21</td>
</tr>
<tr>
<td>RegNetY-12GF</td>
<td>80.3</td>
<td>175.2</td>
<td>75</td>
<td>12.1</td>
<td>51.8</td>
</tr>
<tr>
<td>RegNetY-32GF</td>
<td>80.9</td>
<td>250.2</td>
<td>46</td>
<td>32.3</td>
<td>32.3</td>
</tr>
<tr>
<td>Swin-T [47]</td>
<td>81.3</td>
<td>-</td>
<td>-</td>
<td>4.5</td>
<td>29</td>
</tr>
<tr>
<td>ViT-S [59]</td>
<td>79.9</td>
<td>66.5</td>
<td>207</td>
<td>4.6</td>
<td>22</td>
</tr>
<tr>
<td>Rev-ViT-S</td>
<td>79.9</td>
<td>88.1 × 7.6 ×</td>
<td>1232 × 6.0 ×</td>
<td>4.6</td>
<td>22</td>
</tr>
<tr>
<td>ViT-B [59]</td>
<td>81.8</td>
<td>129.7</td>
<td>95</td>
<td>17.6</td>
<td>87</td>
</tr>
<tr>
<td>Rev-ViT-B</td>
<td>81.8</td>
<td>170.1 × 7.6 ×</td>
<td>602 × 6.3 ×</td>
<td>17.6</td>
<td>87</td>
</tr>
<tr>
<td>RegNetY-8GF</td>
<td>81.7</td>
<td>147.2</td>
<td>91</td>
<td>8.0</td>
<td>39</td>
</tr>
<tr>
<td>CSWin-T [14]</td>
<td>82.7</td>
<td>-</td>
<td>-</td>
<td>4.3</td>
<td>23</td>
</tr>
<tr>
<td>Swin-S [47]</td>
<td>83.0</td>
<td>-</td>
<td>-</td>
<td>8.7</td>
<td>50</td>
</tr>
<tr>
<td>ViT-L</td>
<td>81.5</td>
<td>349.3</td>
<td>26</td>
<td>61.6</td>
<td>305</td>
</tr>
<tr>
<td>Rev-ViT-L</td>
<td>81.4</td>
<td>22.6 × 15.5 ×</td>
<td>341 × 13.1 ×</td>
<td>61.6</td>
<td>305</td>
</tr>
<tr>
<td>MViT-B-16 [18]</td>
<td>82.8</td>
<td>153.6</td>
<td>89</td>
<td>7.8</td>
<td>37</td>
</tr>
<tr>
<td>Rev-MViT-B-16</td>
<td>82.5</td>
<td>66.8 × 2.3 ×</td>
<td>157 × 1.8 ×</td>
<td>8.7</td>
<td>39</td>
</tr>
</tbody>
</table>

slightly higher FLOPs due to the stage-transition (§3.3.1) stages while still being very GPU memory efficient owing to the stage-preserving (§3.3.2) stages.

Increasing memory savings with depth. In Table 1, we observe that our Rev-ViT matches the performance of vanilla ViT to very close fidelity across all model variants (Small, Base and Large) and FLOP regimes. Since the memory used per image is linearly dependent on the depth of the model for vanilla networks (§3.1.2), the memory gains of the reversible model increases as the network scales in depth. Notably, while the Reversible ViT-S already enjoys an impressive memory saving of about 86.8% (equivalent to a 7.6× reduction) with respect to the vanilla ViT-S model, the gain increases further to 15.5× or, about 93.5% memory savings for the Reversible ViT-L model.

Increasing batch size with depth. Conversely, the saved memory can be used to increase the training batch size where we observe a similar trend. While reversible ViT-S model achieves a 6.0× increase in batch size on the ViT-S model, the effect is much more magnified on the ViT-L model where the maximum batch size increases by 13.1× jumping from a small 26 image per batch to 341 images. This is a very desirable trend, since it is the deeper models that hit the GPU memory wall [21].

Further, hierarchical vision transformers such as MViT also enjoy a memory saving of about 56.5% without suffering any significant drop in performance. The memory savings in Rev-MViT are smaller compared to the ViT variants because of the stage-transition blocks in hierarchical models (§3.3.1) that require storing the input activations due to the non-reversible nature of pooling attention, stemming from the feature dimension change [18].

4.2. Video Classification

Settings. We also benchmark Rev-MViT-B models on action classification on Kinetics-400 [35] and Kinetics-600 [5] datasets. All the models are trained from scratch with training recipes adapted from [18].

Results. Table 2 and 3 present the results on action recognition task on the Kinetics-400 [35] and Kinetics-600 [5] respectively. For action recognition, we benchmark our adapted Reversible MViT model and report top-1 performance for both datasets. Similar to the image classification benchmark, we observe that the Reversible MViT models closely match the accuracy for their non reversible counterparts at a fraction of the memory cost.

Increasing video model batch sizes. We note that our adapted Reversible MViT forms a very competitive video
4.3. Object Detection

We benchmark the proposed Rev-ViT-B and Rev-MViT-B models on object detection on MS-COCO [44] as well. All the models are trained on 118K training images and evaluated on the 5K validation images. We take the ViT-B backbone pre-trained on IN and use the standard Mask R-CNN [25] as the detection framework. All models are trained with a standard 3× schedule (36 epochs). For MViT, we integrate the multi-scale backbone with the feature pyramid network [43]. Referring to Table 4 we observe that the Rev-MViT-B model closely matches the AP performance on MViT-B at only 57.6% of the memory cost.

4.4. Ablations

**Stronger Inherent Regularization.** Across different models and datasets, we find that at the same FLOP and parameter specifications, the reversible models tend to have a stronger inherent regularization than their non-reversible counterparts. Hence, training recipes for reversible vision transformers have lighter repeated augmentations, smaller augmentation magnitudes and consequently, higher weight decay. Table 5 shows the effects of these recipe changes on Rev-ViT-B. In addition, we also note the low accuracy of the naïve reversible adaptation (see Figure 3a). We also observe similar effects on Rev-MViT models where a modified training recipe with lighter augmentations and weight decay play a crucial role in matching performance.

Lateral Fusion Strategies. The stage-transition blocks employ residual stream fusion blocks for mixing information between I1 and I2 (§3.3.1). We explore several fusion strategies in Table 6 using a combination of: (A) n×-MLP: Two layer perceptrons with n times the hidden dimension and GELU activations. (B) 0. n dp: n × 10 percent dropout on output activations. (C) Simple operators such as channel-wise maximum of I1 and I2 activations, and channel-wise concatenation of tensor.

Lateral connections in stage-transition stages allow effective information mixing between the residual streams and hence increase network capacity. For example, compared to concatenation in stage-transition blocks (second row), 2×-MLP increases to training accuracy by 1% and also the top-1 performance by 0.5%. However an even heavier strategy, such as 4×-MLP widens the generalization gap and promotes over-fitting while hurting performance. Note that the training accuracy is often lower than the top-1 accuracy.

Re-configuring residual connections. As discussed in §3.2.3, the reversible vision transformer design removes the skip connections that are commonly used inside the Attention and MLP blocks (Figure 2). Specifically, for all of the reversible blocks in Rev-MViT and Rev-ViT, the inputs I1
Reversible training and maximum batch size. (b) Training throughput to larger models by 8x to 36. (c) Maximum training batch size vs. ImageNet Top-1 Acc.

Table 3.

<table>
<thead>
<tr>
<th>Model</th>
<th>Batch Size = 1</th>
<th>Batch Size = 4</th>
<th>Batch Size = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours (without caching)</td>
<td>3.0</td>
<td>4.0</td>
<td>4.5</td>
</tr>
<tr>
<td>Ours (with caching)</td>
<td>3.5</td>
<td>5.0</td>
<td>5.5</td>
</tr>
<tr>
<td>With internal residuals</td>
<td>4.0</td>
<td>5.5</td>
<td>6.0</td>
</tr>
</tbody>
</table>

Figure 3. Ablation Experiments. (a): We observe that (1) Learning without activation caching does not hurt reversible accuracy for Rev-ViT-B of varying depths and (2) Internal residual connections diverge for deeper models. (b): Rev-MViT has higher throughput for higher input resolution and deeper MViT models increasing up to 2.3x at 224 resolution for 80 layers on 16G V100 and up to 3.9x on 40G A100. (c): We benchmark the maximum batch size for Rev-ViT Base (B), Large (L) and Huge (H) and their non-reversible counterparts.

Effect of learning without caching activations. Figure 3a also compares the image classification performance of the Rev-ViT-B architecture trained with and without caching activations. This allows us to disentangle the effect of the proposed residual configurations necessary for reversible vision transformer from any artefacts that might result from learning without caching activations. However, for all depths we notice the Rev-ViT-B performance trained without caching activations to closely track the performance of the same architecture trained with caching. The slight difference at depth 12 results might stem from the training recipe being adapted for the actual Rev-ViT-B architecture trained without activations.

Model size vs. Training Speed. Figure 3b shows the training throughput comparisons for different models sizes at 224 and 320 input resolutions. We note that while for smaller models such as, MViT-B with a depth 12 layers, Rev-MViT-B has a slightly smaller training throughput (98.5 vs. 86.0), the additional re-computation burden of activations is promptly overcome at both higher resolution training as well as for deeper models. In particular, at 224 resolution, the 24-layer and 48-layer Rev-MViT models have similar throughput as the MViT models. At 80 layers, the training throughput of Rev-MViT is higher, increasing up to 2.3x on 16G V100 and upto 3.9x on 40G A100 for 224 resolution than the corresponding vanilla variant. Further, the rate of memory increase vs. depth is much smaller for Rev-MViT, allowing scaling to deeper models without additional infrastructure burden or memory requirements as with gradient checkpointing or model parallelism.

Maximum batch-size. We benchmark the maximum possible batch size for Rev-ViT Base (B), Large (L) and Huge (H) and their non-reversible counterparts in Fig.3c. We extrapolate the trend (denoted by - - - -) to larger models by scaling ViT-L and ViT-H in depth (keeping other model dimensions constant) and benchmark the maximum batch size for their reversible counterparts.

Model size vs. GPU memory footprint. Figure 1 plots the GPU Memory footprint for both Rev-ViT and Rev-MViT family of models as well as for several other prior networks such as MViT [18], ViT [15], ResNets and RegNetY [55]. We note that at fixed GFLOPs, reversible variants are extremely memory efficient going upto 4.5x for MViT and 15.5x for ViT surpassing prior convolutional variants by orders of magnitude.

5. Conclusion

We present Reversible Vision Transformers, memory-efficient reversible architectural adaptations of ViT and MViT models. We benchmark across several tasks, such as image classification, object detection and video classification and across several metrics, such as model complexity, throughput, accuracy and memory usage. Across all specifications, our Rev-ViT and Rev-MViT match the accuracy of non-reversible variants at a tiny fraction of the memory cost while maintaining similar training throughput for smaller models and up to 2.3x higher throughput for larger models. Specifically, we observe that the Rev-ViT and the Rev-MViT models achieve upto 15.5x and 4.5x lighter memory footprint than ViT and MViT models respectively. In future work, we aim to use the proposed reversible model as key design space element for deeper and memory-efficient visual recognition architectures.
References


[18] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In Proc. ICCV, 2021. 1, 2, 5, 6, 7, 8


