

Making Reversible Transformers Accurate, Efficient, and Fast

Tyler Zhu



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-104

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-104.html>

May 11, 2023

Copyright © 2023, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

I am grateful to my advisor, Jitendra Malik, for allowing me to pursue vision research and influencing my understanding of the field.

I would also like to thank Professor Kurt Keutzer for reading my thesis and providing valuable feedback.

I cannot thank my research mentor, Karttikeya Mangalam, enough for his unwavering faith in me and his guidance throughout my master's.

His wisdom, tenacity, and vision have been instrumental in my growth as a researcher, and the lessons I learned will carry me far into the rest of my journey.

I would also like to thank Alvin Wan and Dan Hendrycks, who also advised me throughout my undergrad years and taught me much of what I know about research. Finally, thank you to my friends and family, who have always believed in me and supported me to the end.

Making Reversible Transformers Accurate, Efficient, and Fast

by

Tyler Zhu

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley


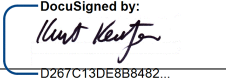
Committee in charge:

Professor Jitendra Malik, Chair

Professor Kurt Keutzer

Spring 2023

The thesis of Tyler Zhu, titled **Making Reversible Transformers Accurate, Efficient, and Fast**, is approved:

Chair	Jitendra Malik		Date	5/11/2023
	Kurt Keutzer		Date	5/9/2023
			Date	

University of California, Berkeley

Abstract

Making Reversible Transformers Accurate, Efficient, and Fast

by

Tyler Zhu

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

The increasing prevalence of a unified architecture for machine learning, i.e. the transformer, raises an important question: can a single architecture really do it all? Simultaneously, the growing size of datasets and deep learning models has made faster and memory-efficient training crucial. One recently proposed line of work is reversible networks, which leverage reversible transformations to perfectly reconstruct inputs from outputs while requiring very minimal changes to existing architectures. In this work, we present an in-depth analysis of reversible transformers and demonstrate that they can be more accurate, efficient, and fast than their vanilla counterparts. We introduce a new method of reversible backpropagation which is faster and scales better with memory than previous techniques, and also demonstrate new results which show that reversible transformers transfer better to downstream visual tasks.

Contents

Contents	i
List of Figures	ii
List of Tables	iv
1 Introduction	1
2 Background	3
2.1 Reversible Networks	3
2.2 Reversible Transformers	4
3 Fast and Efficient Reversible Transformers	6
3.1 Introduction	6
3.2 Parallelized <u>R</u> eversible Back <u>p</u> ropagation	7
3.3 Results	10
4 Accurate Reversible Transformers	14
4.1 Introduction	14
4.2 CLEVR Evaluation	16
4.3 Detection	18
4.4 Conclusion	19
5 Conclusion	21
Bibliography	22

List of Figures

2.1	Illustration of the Reversible Transformation with arbitrary functions F, G . See Eq. (2.3) for the mathematical definition.	4
3.1	Parallelized Reversible backpropagation. PaReprop (bottom) parallelizes the activation re-computation stage of block $i - 1$ (green blocks) with the gradient computation stage of block i . Note that reversible training drastically alleviates training memory burden of vanilla networks (top vs. middle rows) but introduce the additional burden of activation re-computation in the backward pass (see [25]). PaReprop further alleviates this re-computation burden with parallelization, thereby making reversible architecture a practical choice for deep transformer training.	8
3.2	Reversible Swin. [25] introduces the Reversible ViT and MViT architectures. Following the same principles, we introduce the Reversible Swin architecture and benchmark all the three reversible architectures with PaReprop. We showcase (a) a typical Reversible Swin block, as well as (b) a downsample block for processing at multiple hierarchies of scale.	9
3.3	PaReprop Training throughput vs. Batch size across model architectures and sizes. One s.d. error bars are shown. As mentioned in Section 3.3, all accuracies are the same as original methods.	10
3.4	PaReprop Training Throughput across Sequence length and Batch Sizes. Comparison of our method on RoBERTa, a language transformer. One s.d. error bars are shown.	11
3.5	PaReprop Memory Vs. Batchsize. Our method, PaReprop, as well as standard reversible backprop, Reprop, use comparatively much less memory than nonreversible approaches (Backprop). Our memory increase to achieve our speedups is very minimal compared to the overall memory savings	12
4.1	An example of a CLEVR picture we would generate. In this example, both attributes X , a rubber green sphere, and Y , a yellow metal cube, are present. .	15
4.2	RevViT improvement over ViT. RevViT offers significant improvements over ViT across different correlations of features in our pretrain dataset, as well as over different amounts of data as depicted by the legend.	17

- 4.3 **RevViT improvement over ViT.** Comparison of Rev-ViT on the transfer dataset over ViT. Reversible architectures show a consistent improvement over vanilla vision transformers across correlations and data size in the pretraining set. 18

List of Tables

4.1	Details of each of the datasets used in our experiments. Here, the attribute X is a rubber green sphere, while attribute Y is a metal yellow cube. $\mathbb{P}(X)$ is 0 in the transfer dataset because X should not be present, so we can observe how well other cues are picked up on.	16
4.2	Reversible backbone features transfer better for object detection. We use a frozen ImageNet-1k pretrained model as our backbone for ViTDet, and observe how well the architecture does on top. The amount of parameters and FLOPs used in both is the same, and ours does better in all settings, and especially on Small models.	19

Acknowledgments

There are many mentors whom I have to thank for their support along my research journey. I would not be here without any of their belief in me. First, I am grateful to my advisor, Jitendra Malik, for allowing me to pursue vision research and influencing my understanding of the field. I would also like to thank Professor Kurt Keutzer for reading my thesis and providing valuable feedback. I cannot thank my research mentor, Karttikeya Mangalam, enough for his unwavering faith in me and his guidance throughout my master's. His wisdom, tenacity, and vision have been instrumental in my growth as a researcher, and the lessons I learned will carry me far into the rest of my research journey.

I would also like to thank Alvin Wan and Dan Hendrycks, who also advised me throughout my undergraduate experience and taught me much of what I know about research. Finally, I would like to thank my friends and family, who have always believed in me and supported me to the end.

Chapter 1

Introduction

The field of deep learning has made great strides in recent years on the back of large-scale models. In particular, the transformer [29] has become the de facto architecture for many fields, and has been shown to be effective on a wide variety of tasks, such as image recognition [9], language modeling [3], and speech recognition [1]. Given the ubiquity of the transformer, it is natural to wonder if a single architecture really can do it all. Many recent works have proposed significant modifications to the transformer to improve certain properties. For example, many works in language have tried to improve the efficiency of attention like Linformer [30], Performer [6], and Sinkhorn transformer [28], but none of them have been able to match the vanilla transformer in pure accuracy when put to the test of scale.

In vision, the vision transformer [9] has been shown to be able to exceed the performance of convolutional neural networks on image recognition tasks, while also being more efficient and scalable. Hierarchical improvements such as Swin transformer have been proposed to improve the performance of vision transformers, especially on downstream tasks where better features will be appreciated [23]. However, recent works have found that vanilla transformers can be a competitive alternative to these hierarchical approaches for downstream tasks like object detection [20], suggesting that the pure vanilla transformer may be a more general architecture than previously thought.

As these models become larger, it becomes equally as important to improve the efficiency of these models so that they can be deployed in real-world applications. Models need to be able to run on smaller amounts of compute and run faster. When models are at the huge scale that they are today, small speedups can lead to large gains over the course of the many epochs and training *weeks* that are required to train these models. Many architectures which aren't *pure* transformers tend to trade off efficiency in training time for other properties such as accuracy or generality.

In this work, we follow the wisdom of keeping things simple and as close to a pure transformer as possible. We are motivated by memory efficiency to consider a very simple modification to the transformer architecture which maintains the same FLOPs and parameters, but requires much less memory to train. As a consequence, we are able to train our models faster than even plain transformers while maintaining the same accuracy.

To accomplish this, we borrow from the literature of generative flow-based methods and introduce *reversible transformations* into the transformer architecture, creating a *reversible transformer*. Reversible networks have been investigated in the past, especially in vision [13, 25], where they have shown to have great memory savings, and thus runtime speedups. We improve upon these findings and propose two new advancements for reversible transformers. The first is a new method for parallelizing reversible backpropagation which is faster and scales better with memory than previous techniques which we call *PaReprop*, or *Parallelized Reversible backpropagation*. The second is a new investigation into how the features learned by reversible transformers transfer better than vanilla transformers to downstream visual tasks. We demonstrate that reversible transformers can be more accurate than vanilla transformers on image recognition tasks, and that they can transfer better to downstream tasks like object detection. We hypothesize that this is due to the lossless nature of reversible transformers, which allows for more information to be preserved in the features.

In these two methods, we not only show how we can make reversible transformers more fast and efficient, but also how they can be more accurate than vanilla transformers. We hope that this work will inspire future work to investigate reversible transformers further as a convincing alternative to transformers that has many desirable properties without sacrificing efficiency.

Chapter 2

Background

2.1 Reversible Networks

Reversible architectures are a type of neural network architecture based on NICE [7, 8], which was an early model for generative flow-based image generation [15, 17]. At its core, NICE uses a transformation which can perfectly recover its inputs from its outputs by being wired in a specific manner. This was useful for generative modeling because it allowed for tractable density estimation, which is a requirement for generative models.

The Reversible Transformation

A reversible transformation T maps inputs I_1 and I_2 to outputs O_1 and O_2 in a manner which may not have an analytical inverse, but can still be inverted. We will utilize intermediate functions F and G , which are not necessarily invertible.

The reversible transformation is composed of two transformations T_1 and T_2 in sequence. The first is defined as

$$\mathbf{I} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \xrightarrow{T_1} \begin{bmatrix} I_2 + F(I_1) \\ I_1 \end{bmatrix} = \begin{bmatrix} O_1 \\ O_2 \end{bmatrix} := \mathbf{O}, \quad (2.1)$$

which admits an inverse T_1' by applying $F(\cdot)$ to I_1 to recover I_2 .

The second transformation is defined as

$$\mathbf{I} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \xrightarrow{T_2} \begin{bmatrix} I_1 + G(I_2) \\ I_2 \end{bmatrix} = \begin{bmatrix} O_1 \\ O_2 \end{bmatrix} := \mathbf{O}, \quad (2.2)$$

which admits an inverse T_2' by applying $G(\cdot)$ to I_2 to recover I_1 . Together, we can compose these transformations to get $T = T_2 \circ T_1$, resulting in

$$\mathbf{I} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \xrightarrow{T} \begin{bmatrix} I_1 + G(I_2 + F(I_1)) \\ I_2 + F(I_1) \end{bmatrix} = \begin{bmatrix} O_1 \\ O_2 \end{bmatrix} := \mathbf{O} \quad (2.3)$$

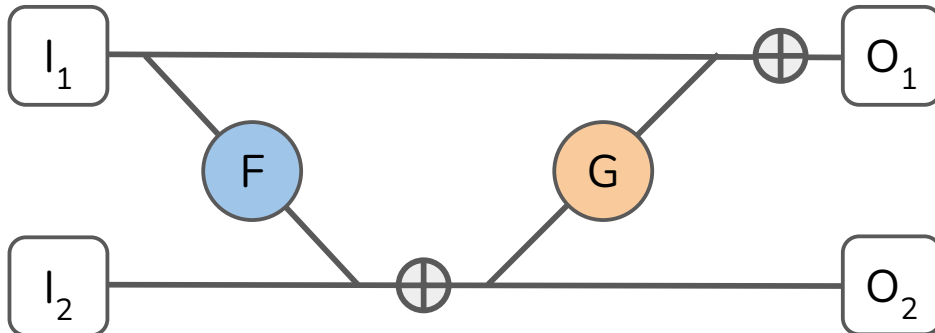


Figure 2.1: **Illustration of the Reversible Transformation** with arbitrary functions F, G . See Eq. (2.3) for the mathematical definition.

This process is illustrated in Figure 2.1. It requires one call of $F(\cdot)$ and $G(\cdot)$ to compute either T or an inverse $T' = T'_1 \circ T'_2$, so both the forward and the backward pass require the same computational cost. However, a key benefit is that we don't need either F or G to be analytically invertible to do this inversion. This transformation is the backbone of much of the work on reversible networks.

Reversible ResNet [13] is a type of reversible architecture that uses the NICE invertible transformations to enable memory-efficient image classification. Other researchers have built upon this idea by proposing improved reversible CNN models using ODE characterizations [4, 19, 26], momentum [19, 26], and several other improvements [14, 11, 2, 27]. Recently, reversible networks have also been adapted to core NLP tasks in Reformer [18] and to several core vision tasks in RevViT [25]. Crucially, RevViT [25], under very strict parity constraints on parameters, FLOPs and activation sizes of the proposed models, shows reversible models to an equivalently powerful class of models as vanilla transformer but with the added benefit of extremely memory-efficient training.

2.2 Reversible Transformers

Reversible Transformers ([18, 25]) are a class of models which are composed of reversible transformations. The key benefit of this paradigm is that inputs can be recomputed solely from the outputs, removing the need to store intermediate activations. This allows for the model to be trained with a smaller memory footprint and enables the use of larger batch sizes, which can lead to speedups.

Reversible Vision Transformers

In the case of a Reversible Vision Transformer (RevViT) [25], the model is built out of reversible blocks which are composed of an attention block for $F(\cdot)$ and an MLP block for $G(\cdot)$. As all the reversible blocks are connected consecutively, the entire model is reversible. This means that during our forward pass, we do not need to store any activations. In the backward pass, we simply need to recompute the activations of the current block using the output, and then backpropagate to recover our gradients and update our weights. We can then delete the activations of the current block, and repeat this process for the next block.

Following this setup, RevViT showed that reversible transformers could perform on par with standard vision transformers, while using significantly less memory. In fact, in constrained memory settings or for very large models, RevViT offers significant speedups due to the efficiency unlocked by using larger batch sizes.

Chapter 3

Fast and Efficient Reversible Transformers

3.1 Introduction

Reversible vision transformers [25] is one recent result which offers a promising approach to improving the efficiency of large models by decoupling the memory needed from the depth by using reversible activations. By doing so, they are able to maintain top performance on various visual tasks, while also requiring less memory.

The framework of reversible transformations however offers a further tradeoff where for a tiny amount of additional memory, we can improve our throughput by a significant amount using parallelization. This takes advantage of the independence between the recomputations and gradient updates in the backward pass which allows them to theoretically be computed simultaneously. From this observation, we introduce a method for parallelizing a reversible backpropagation which we call *PaReprop*, or *Parallelized Reversible backpropagation*. Our method is general enough to be applied to any reversible architecture, and we demonstrate this by testing on a wide variety of reversible architectures, hardware, and memory settings, and show that we can achieve significant speedups in practice. In summary, we make the following contributions:

1. We propose a novel method for parallelizing reversible backpropagation which is compatible with modern auto-differentiation packages like PyTorch.
2. Our method achieves significant speedups across a diverse set of model families, data modalities, model sizes, and training batch sizes. We increase training throughput for all of our models while maintaining the same accuracy as the original model, up to 20% on models which are the least “pure”, i.e. with many extra operations such as shifted window attention.
3. Using PaReprop scales better on throughput with memory than standard reversible backpropagation. In particular, PaReprop leads to more favorable memory vs. through-

put trade-offs, i.e. our method can achieve higher throughput at any given threshold of memory used.

3.2 Parallelized Reversible Backpropagation

We begin with a review of the reversible transformer architecture as proposed in [25]. We introduce the vanilla memory-efficient reversible training algorithm (Section 3.2) and its application to training modern transformers. Then, we present **PaReprop**, our proposed procedure for speeding up reversible backpropagation with parallelized activation and gradient computation (Section 3.2).

Reversible Transformers

Reversible Transformers ([12, 25]) are a family of memory-efficient models based on the idea of the reversible transformations.

We discussed the details of the reversible transformation in Section 2.1. We simply need the property that they allow us to perfectly recompute any input \mathbf{I} from its output \mathbf{O} , which can be used at the granularity of transformer blocks.

This means that during our forward pass, we do not need to store any activations. In the backward pass, we simply need to recompute the activations of the current block using the output, and then backpropagate to recover our gradients and update our weights. We can then delete the activations of the current block, and repeat this process for the next block, as shown in normal Reversible Backpropagation, or Reprop, in Figure 3.1.

However, the backward pass of block N , i.e. the gradient update, is not needed to recompute the activations of block $N - 1$. Therefore, if we can hide the forward pass of block $N - 1$ within the backward pass of block N , we can theoretically speedup our computation on par with that of normal backprop. This is our key insight, which we will now present in detail.

Parallelizing with Two Streams

Our method’s key contribution is both theoretical and practical. The first is illustrated in Figure 3.1, where our PaReprop method is able to parallelize the backward pass of a normal reversible backprop so that it takes nearly the same time as vanilla backprop. We do this by performing the gradient update for block N at the same time as the activation recomputation for block $N - 1$, as there is no dependence once we have the activations for block N .

However, achieving this parallelization is rather tricky. Standard autodifferentiation frameworks like PyTorch use CUDA streams by default to maintain sequential ordering for GPU operations. Our method extends this by maintaining multiple CUDA streams to parallelize operations over. However, these streams enforce that forward and backward passes occur on the same stream, which causes issues if we implement PaReprop naively by keeping

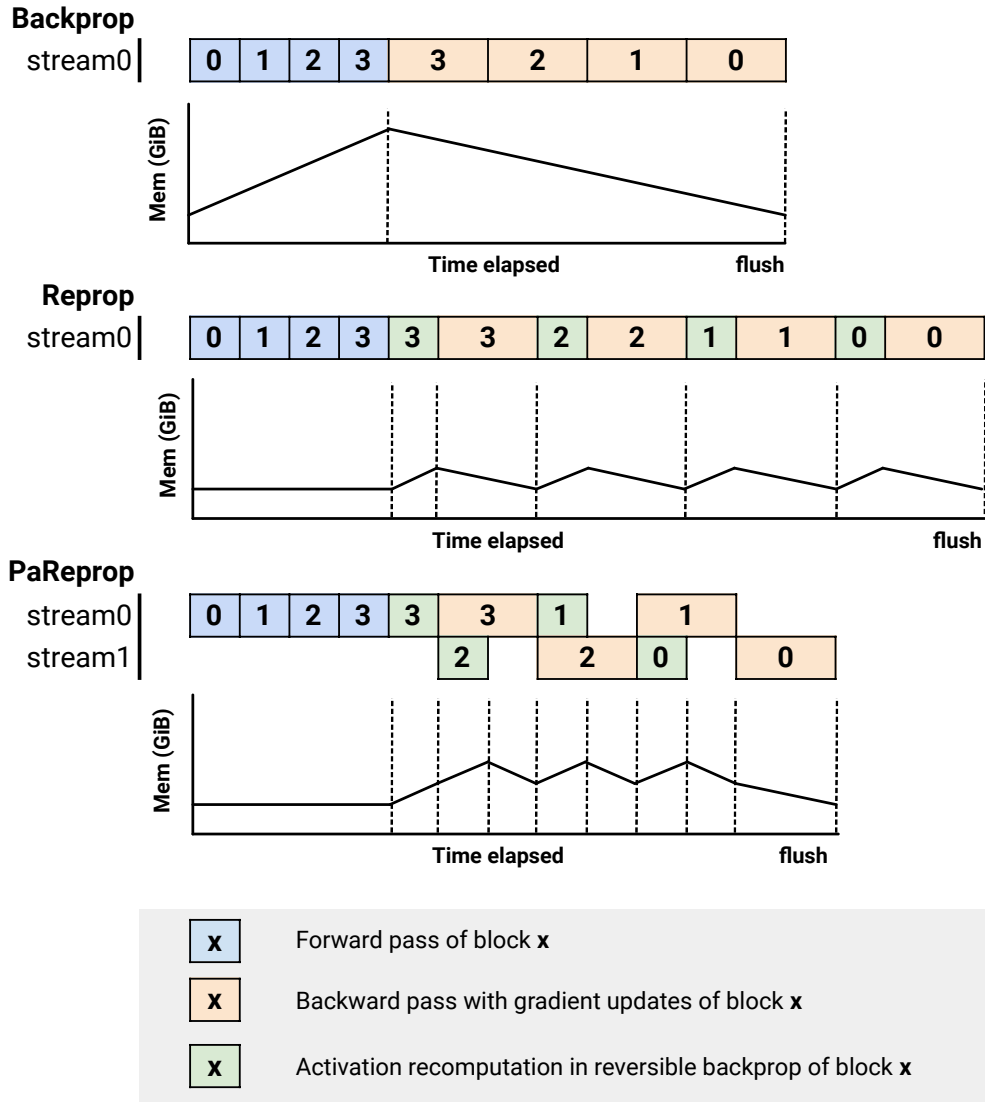


Figure 3.1: **Parallelized Reversible backpropagation**. PaReprop (bottom) parallelizes the activation re-computation stage of block $i-1$ (green blocks) with the gradient computation stage of block i . Note that reversible training drastically alleviates training memory burden of vanilla networks (top vs. middle rows) but introduce the additional burden of activation re-computation in the backward pass (see [25]). PaReprop further alleviates this re-computation burden with parallelization, thereby making reversible architecture a practical choice for deep transformer training.

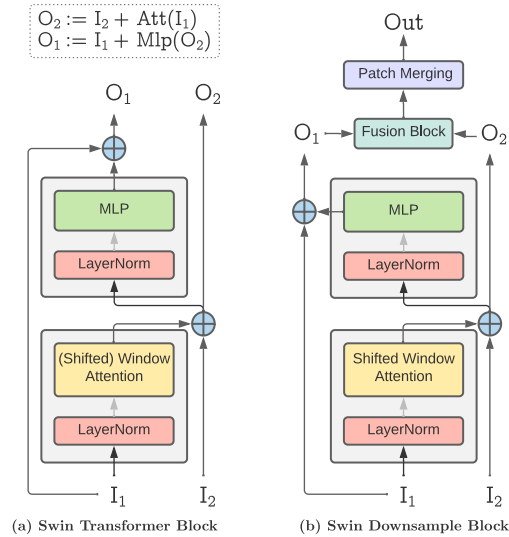


Figure 3.2: **Reversible Swin.** [25] introduces the Reversible ViT and MViT architectures. Following the same principles, we introduce the Reversible Swin architecture and benchmark all the three reversible architectures with PaReprop. We showcase (a) a typical Reversible Swin block, as well as (b) a downsample block for processing at multiple hierarchies of scale.

one stream for the activation recomputation and another for the gradient updates. This necessitates our alternative computation scheme to prevent greater overhead.

Another problem is that PyTorch is unable to free memory efficiently in parallel processes as there is no asynchronous implementation of CUDA-free yet. Thus, it requires a costly CUDA-free operation which synchronizes our streams and thus slows our process down significantly. In practice, it’s most beneficial to run our PaReprop method at anywhere from 33% to 50% of the empirical maximum batch size so that we don’t hit this trap.

Reversible Swin and RoBERTa

In order to demonstrate the generality of our method to other architectures, we propose two novel reversible architectures based on established models: Swin Transformer and RoBERTa. Figure 3.2 shows our modifications to the original Swin architecture.

We follow suit with the original reversible ViT authors and choose to keep the Attention blocks and MLP blocks as our reversible subfunctions. We also demonstrate how we handle multiscale features in the architecture by utilizing a fusion block (either averaging or a simple MLP) before the usual patch merging layer. For RoBERTa, we follow similar steps and rewire the architecture slightly so that our residual connections are free of LayerNorms and thus can help recompute our activations.

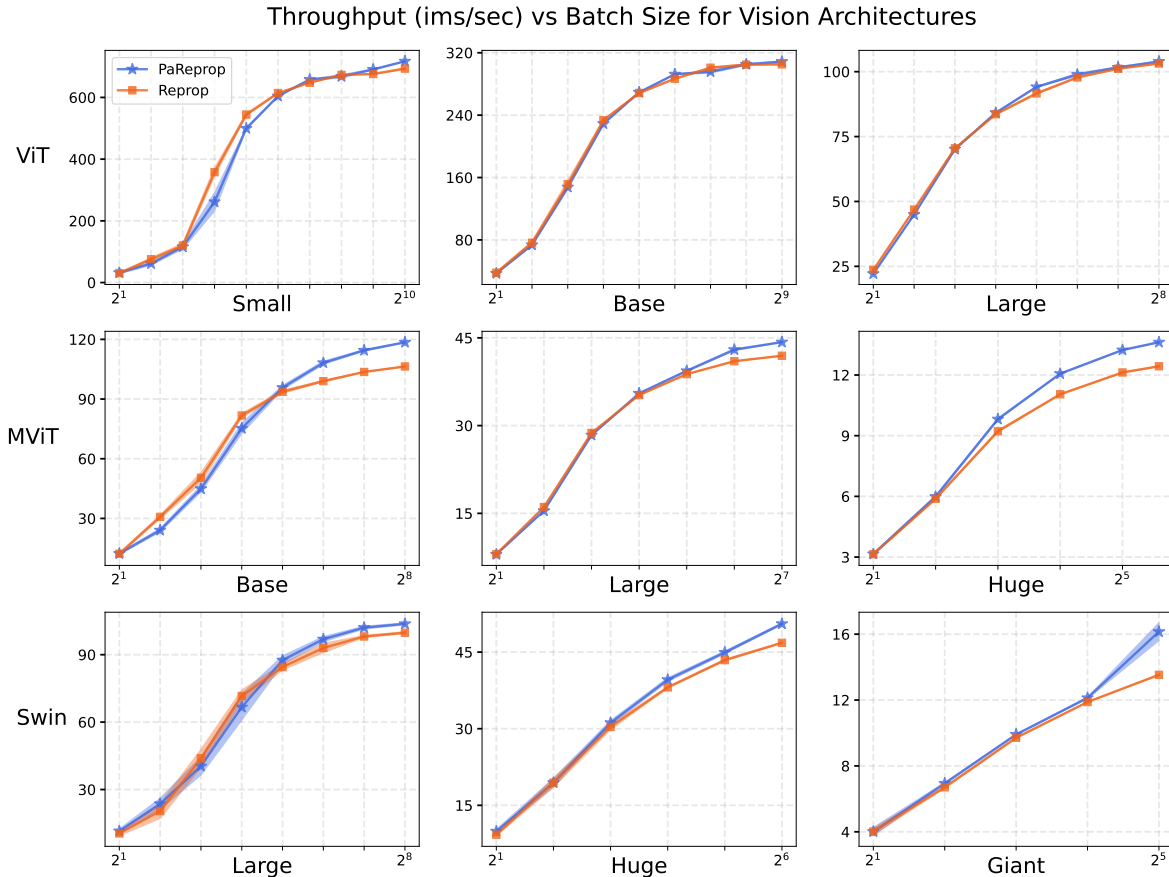


Figure 3.3: **PaReprop Training throughput vs. Batch size** across model architectures and sizes. One s.d. error bars are shown. As mentioned in Section 3.3, all accuracies are the same as original methods.

3.3 Results

In this section, we present our experimental results of our proposed method, denoted as PaReprop, in comparison with the original reversible ViT, denoted as Reprop. We analyze our method over the choice of backbone size (from 5.5M to over 2B parameters), architecture class (ViT [9], Swin [23] and MViT [9, 10], RoBERTa [22]), data modalities (Vision and NLP), GPU training memory allocated, input sequence length (for RoBERTa) and batch size. The primary metric we are concerned with is training throughput (images/sec or sequence/sec), which is the number of images (or sequences) we can process per second, as our measure of speed.

We do not modify the underlying algorithm at all but simply propose a faster implementation, which means our methods achieve the same performance as the original reversible method, so we focus on analyzing the speedup of our method instead. All of our results are

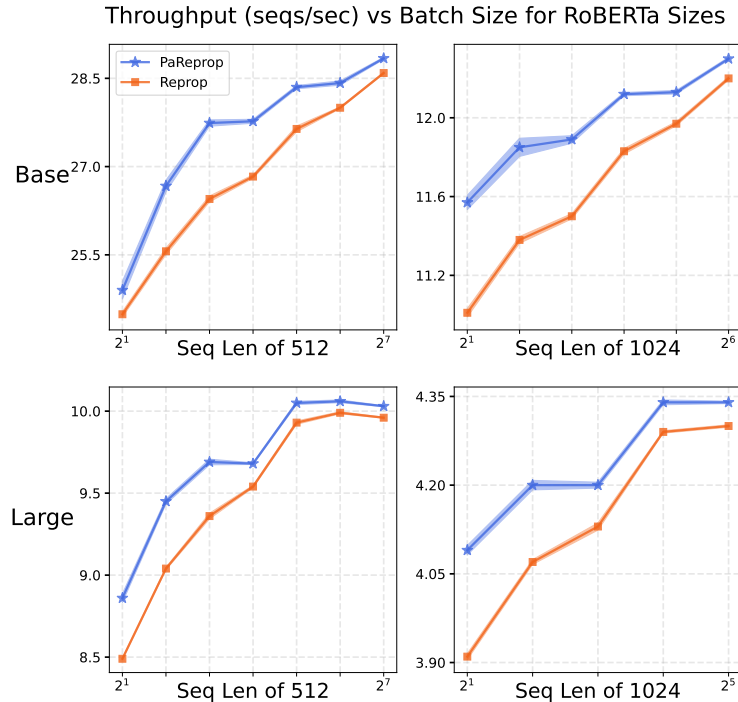


Figure 3.4: **PaReprop Training Throughput across Sequence length and Batch Sizes.** Comparison of our method on RoBERTa, a language transformer. One s.d. error bars are shown.

run on a single NVIDIA A100 GPU with 80GB of memory w/ AdamW as our optimizer. Our results were similar on an NVIDIA RTX 2080 Ti as well as with SGD.

PaReprop Training Throughput is Better

In the first experiment, we compare our PaReprop method with the original Reprop method used in the original reversible ViT. We compare the top throughput achieved over batch sizes of $\{1, 2, 4, \dots, 256, 1024\}$ (as allowed by memory) for each backbone. In these cases, we run on standard image classification, but our results will hold over any choice of task (video understanding, etc.). We see that across three different architecture families (ViT, MViT, Swin) and three choices of model sizes, our method outperforms Reprop, in some cases drastically.

Vision Transformers show a mostly matched throughput between PaReProp and standard reversible backpropagation (Figure 3.3, top row). We find that because ReProp can already utilize a large batch size, the GPU utilization is quite high and the PaReprop kernels are unable to run in parallel.

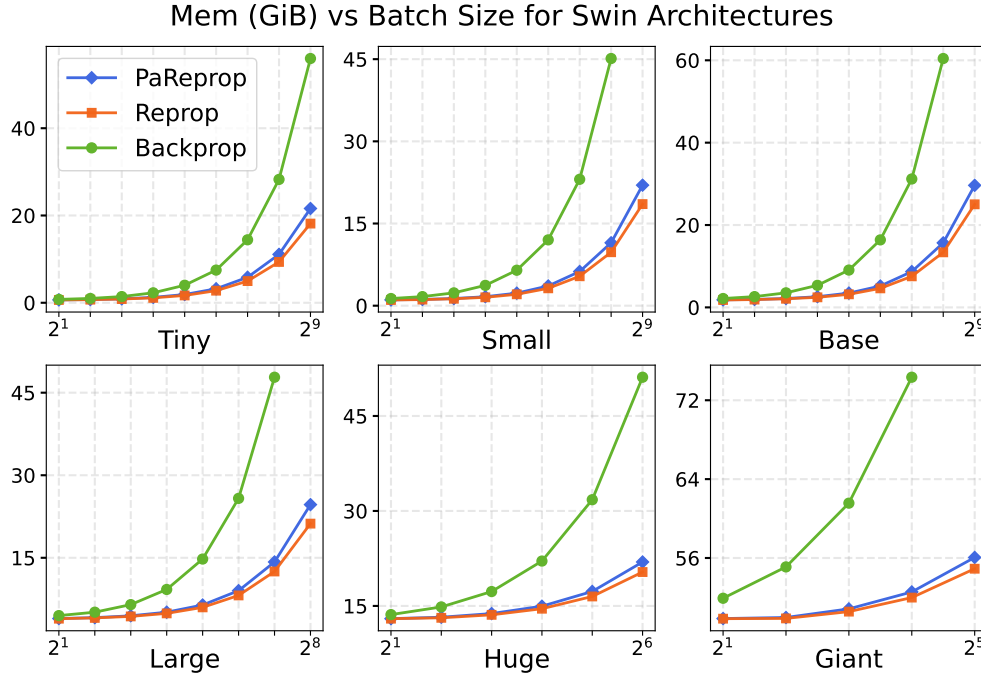


Figure 3.5: **PaReprop Memory Vs. Batchsize**. Our method, PaReprop, as well as standard reversible backprop, Reprop, use comparatively much less memory than nonreversible approaches (Backprop). Our memory increase to achieve our speedups is very minimal compared to the overall memory savings

Hierarchical Transformers enjoy a much more pronounced benefit with PaReprop such as, Multiscale Vision Transformer (Figure 3.3, middle row) and Swin Transformer (Figure 3.3, bottom row). Hierarchical transformers have non homogeneous architectures that allow PaReprop to hide the recomputation latency. They have several small kernels that can be parallelized more effectively, which leads to significant speedups of 9.8% on the largest MViT-H and a 19.3% increase on largest Swin-G, consistently outperforming the standard reversible backpropagation method (Reprop). This shows that PaReprop provides significant speedup for models that are more specialized for vision.

NLP Transformers. Finally, we also clearly demonstrate PaReprop gains on the natural language modality. Reversible models have also successfully been applied to language tasks such as Reformer. Following [25], we extend RoBERTa [22] to Rev-RoBERTa and provide throughput benchmarking results on sentiment analysis. Note that as shown in Rev-ViT [25] using a simple reversible rewiring of the model maintains the original accuracy of the vanilla model. Figure 3.4 shows our method outperforming the original Reprop by large amounts across both choices of model size (Base and Large) and sequence lengths (512 and 1024).

Training GPU Memory Trends

We also investigate the effect of memory on our method (Figure 3.5). Specifically, we compare the memory used by our method with the original Reprop and the vanilla backprop, and show that our method is more memory efficient. As shown in the plots, using any kind of reversible backpropagation offers memory savings of up to almost 3x in some cases, which allows us to significantly extend the batch sizes that we can use. In these scenarios, using our parallelized reversible backpropagation requires only an extra fraction of the small amount of memory being used to maintain parallelization and allows us to achieve higher throughputs. This finding is consistent across all model architectures we tested, but we illustrate most of our findings on Swin in Figure 3.5 for simplicity.

Chapter 4

Accurate Reversible Transformers

4.1 Introduction

While we have primarily explored reversible transformers from an efficiency perspective so far, the reversible structure also has implications for what kind of features are learned. In the modern day, to tackle more difficult core vision problems such as segmentation and detection, we began to use our pretrained models from image classification as initializations for detecting low-level features. As a result, the features learned by our models during pretraining are crucial for downstream performance, so models became larger as we used increasingly more data to train them.

All these findings resulted in the common wisdom that pretrained supervised ImageNet accuracy was the gold standard for achieving the best performance in most cases. Thus, the focus of the field began shifting to discovering the best architecture for image classification which could then be transferred downstream to other core vision problems. However between all of these architectures, the golden truth that accuracy is the determining factor for downstream performance still holds true.

Reversible networks, on the other hand, offer a different perspective on this commonly held belief. Their reversible design allows them to recover inputs perfectly from outputs during the network, which means that they are lossless during the main network. This leads us to wonder if this losslessness could lead to more general features that could be used for downstream tasks.

We explore these threads with a few experiments to understand how reversible vision transformers can perform differently on downstream transferred tasks. We first ablate reversible networks on a variety of large synthetic datasets that we generated with CLEVR [16]. We perform a large suite of experiments to understand how useful the features reversible networks learn during pretraining are for transferring to different tasks. Then we present some of our results on object detection with single-scale backbones. These experiments provide evidence that reversible vision transformers, through their losslessness properties, produce more transferable features.

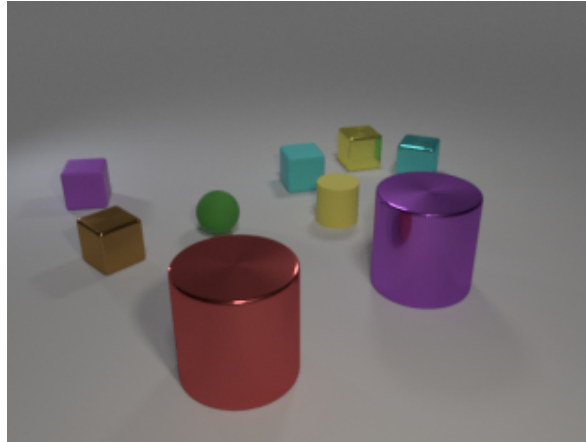


Figure 4.1: An example of a CLEVR picture we would generate. In this example, both attributes X , a rubber green sphere, and Y , a yellow metal cube, are present.

CLEVR

CLEVR [16] is a synthetic dataset generator using Blender which can generate a scene with multiple objects with different attributes. Initially proposed for visual reasoning, the high customizability of CLEVR lends itself easily for testing in a controlled setting. An example of the dataset is shown in Figure 4.1, where multiple objects of different shapes, colors, sizes, and material are located throughout a scene. In our case, we use it as a way to control for the presence of objects which a model may pick up on for identifying features, and seeing how explicit supervision on those objects affects the downstream performance when transferring.

Object detection

Object detection is one of the most core computer vision tasks present today. While most works offer different approaches such as region-proposal based detection or single shot detection, all of them rely on multi-scale backbones, whether they are CNNs or hierarchical transformers like Swin Transformer ([21, 5, 24]).

Recently, ViTDet [20] uses a plain, non-hierarchical backbone in the plain ViT as a backbone for obtaining single-scale feature maps, and simply adding feature pyramid networks on top is enough to obtain competitive results. We use this framework for testing our models as we want to observe Rev-ViT at its full losslessness, with as many reversible modules as possible. Multi-scale blocks introduce non-reversible stages, so we refrain from investigating those architectures.

Type	Examples	$\mathbb{P}(X)$	$\mathbb{P}(Y)$	$\text{Corr}(X, Y)$
Pretrain	150k	0.51	0.51	0.80
Pretrain	150k	0.49	0.64	0.30
Pretrain	150k	0.56	0.56	-0.02
Transfer	50k	0.00	0.56	n/a

Table 4.1: Details of each of the datasets used in our experiments. Here, the attribute X is a rubber green sphere, while attribute Y is a metal yellow cube. $\mathbb{P}(X)$ is 0 in the transfer dataset because X should not be present, so we can observe how well other cues are picked up on.

4.2 CLEVR Evaluation

We first evaluate both reversible and vanilla architectures on synthetic datasets created using CLEVR [16]. We describe our set up in Section 4.2, then detail our experiments and results in Section 4.2. In all, we find that reversible models outperform vanilla models across all of the settings we tested on.

Setup

Our experimental setup is as follows. The goal is to do 0/1 presence classification of if an object with a desired attribute is present in a scene or not. This will help us understand how well reversible vision transformers can pick up on other features while primarily being trained towards its main one.

Datasets We have two styles of datasets: a *pretrain* dataset (pt) and a *transfer* dataset (tf). The pretrain dataset consists of training on the 0/1 objective of if attribute X is present in a scene (for our cases, X is a rubber green sphere). We especially balance our dataset so the classes are equal, i.e. about 50/50. In each scene however, there is also an attribute Y which may or may not be present (in our cases, Y is a metal yellow cube). This is balanced according to different methods which we will discuss later. The transfer dataset then consists of entirely new images where the attribute X is never present in any of the scenes, and the attribute Y is now balanced for.

The purpose of this pretrain dataset is to provide the networks with other objects and features to pick up on which aren't explicitly being supervised. Our transfer dataset then evaluates how useful those other cues were for transfer to the downstream objective.

Training Procedure We first train our networks on the pretrain dataset to learn to detect the presence of an object with attribute X purely, i.e. without telling our model to look for

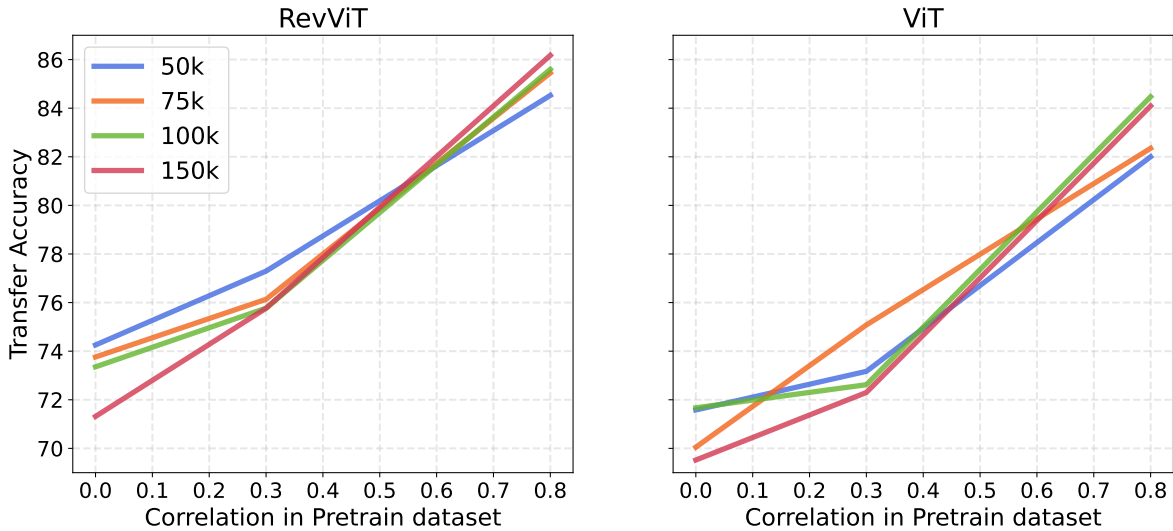


Figure 4.2: **RevViT improvement over ViT.** RevViT offers significant improvements over ViT across different correlations of features in our pretrain dataset, as well as over different amounts of data as depicted by the legend.

attribute Y . Then we observe how well our models transfer to detecting attributes Y with both the backbones frozen and just tuning the head projections, i.e. a linear probe.

To observe how the presence of attribute Y in pretrain scenes affects our models ability to transfer later, we also vary the correlation ρ between the indicator variables measuring presence of attributes X and Y . We generate three datasets of 150k images each with correlations of around $\rho = 0, 0.30, 0.80$. This allows us to vary how strong these relationships are to determine what our models pick up on. More specific details about our datasets are included in 4.1.

For our model, we used a vanilla ViT with its capacity adjusted in order to be suitable for the task at hand. Our input images are received at an 80×80 resolution, and we train on a standard supervised 2-class classification setup. We set our patch size to be 5, depth to be 4, number of heads to be 8, and embedding dimension of 256, and dub this ViT to be ViT-Micro (or ViT-M). During pretraining, we train for 40 epochs at a learning rate of 0.0003 with a single cycle cosine annealing policy using AdamW. We forgo standard augmentations as we are categorizing based on color and other important texture-based attributes, so they could potentially change our classes.

Results

We show our primary results in Figure 4.2. First, we pretrain both RevViT and ViT on the three datasets we created as outlined in Section 4.2. Then we freeze our models and

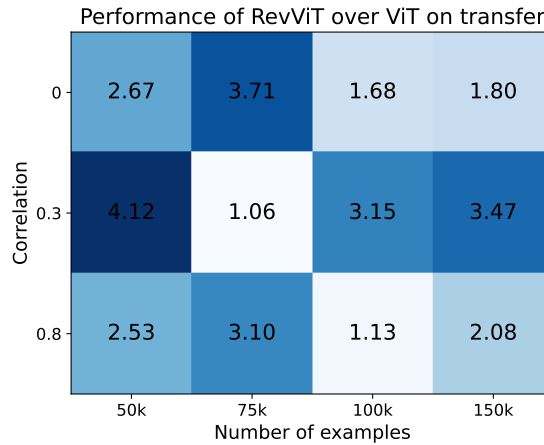


Figure 4.3: **RevViT improvement over ViT**. Comparison of Rev-ViT on the transfer dataset over ViT. Reversible architectures show a consistent improvement over vanilla vision transformers across correlations and data size in the pretraining set.

re-initialize the head in a linear probe. This allows us to fully ablate the performance of the features found by the model during pretraining and understand its effect.

For our different settings, we vary two parameters. The first is the correlation between our pretrain attribute X and the transfer attribute Y . A lower correlation will lead to a lower transfer accuracy, as we expect, while a higher correlation naturally also leads to a higher transfer accuracy. The second is the size of the pretrain dataset. We vary the fraction of the dataset used by different amounts in $(\frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1)$.

As we can see, our RevViT architecture performs better overall across each of the settings and across trends. In Figure 4.3, we more finely break down the amounts by which we outperform ViT. We observe our largest gains primarily around the 0.3 correlation datasets, as well as with less examples seen. This suggests that RevViT can perform well with less data and pick up on useful features more efficiently.

4.3 Detection

We now explore our idea on more complex and core vision tasks, such as object detection and segmentation. As our backbones are primarily single-scale, we utilize the recent method outlined in ViTDet [20] to use our plain vision transformers as the backbone for such tasks.

Setup

For our setup, we follow similar settings to the original paper. We use the stated hyperparameters by default as well as the implementation provided in detectron2, and only add

Type	Model Size	Box AP	Mask AP
Base	Reversible	29.90	28.00
Base	Vanilla	29.00	26.96
Small	Reversible	26.69	24.89
Small	Vanilla	21.20	19.745

Table 4.2: **Reversible backbone features transfer better for object detection.** We use a frozen ImageNet-1k pretrained model as our backbone for ViTDet, and observe how well the architecture does on top. The amount of parameters and FLOPs used in both is the same, and ours does better in all settings, and especially on Small models.

the ability to freeze our backbones in order to isolate the effects of the architecture from pre-training. We initialize from the publicly available ImageNet-1k trained checkpoints for both ViT and RevViT, which have identical accuracies on the pretraining task, i.e. ImageNet image classification.

We test all of our results on MS COCO, and measure both the box AP and the mask AP. As we freeze the entire backbone, we do not add in global propagation or convolutional propagation for information propagation, nor do we add in window attention as is done in ViTDet. Finally, our RevViT is also parameter matched with the ViT backbones we use, and has about the same amount of FLOPs as well.

Results

Our results from frozen backbone training are shown in Table 4.2. We see that overall our method does better than the baseline approach, especially all the small architecture size. On base size architectures, we find a 1 mAP increase overall in both box and mask AP, while in small size architectures we find around 5-5.5 mAP increase there.

These results further suggest that reversible ViTs offer more transferable features especially on tasks which require complex understanding, especially on larger resolution tasks that also have more knowledge embedded in them.

4.4 Conclusion

In this section, we explore the possibilities of using reversible architectures as a competitive alternative for transfer learning. Due to the structure of reversible transformations, they are inherently more lossless than their vanilla counterparts due to being able to perfectly reconstruct the inputs from the outputs of the reversible blocks. In particular, we find that in both our controlled setting of CLEVR, as well as a real world example of object detection, the features learned by RevViT are better for downstream performance than those of a vanilla

CHAPTER 4. ACCURATE REVERSIBLE TRANSFORMERS

20

vision transformer. We hope that this encourages more people to investigate this line of work and further understand the many hidden benefits of using reversible models.

Chapter 5

Conclusion

In this work, we described a recent trend of work utilizing reversible transformers. We first described the core idea of reversible transformers, and then described our work, PaReprop, to make them faster and more efficient. We introduced a fast parallelized reversible backpropagation algorithm that allows us to alleviate the additional computation burden of reversible models by parallelizing the activation recomputation in the backward pass with the gradient calculations itself. This method leads to an up to 20% speedup in training throughput for some models, and in general speeds up reversible models across a large range of architectures, domains, and model sizes.

Then we followed this up with a further investigation into the implications of this slight difference in architecture. Notably, the reversible nature of these models leads to a natural inquiry as to whether it helps them process large-scale features better and thus more general representations that could help with downstream tasks. Our initial foray into this found that reversible models are more accurate than their irreversible counterparts in a carefully constructed setting measuring feature transfer. This also was replicated when we identified how these models performed on object detection with a single-scale backbone. We found that our reversible models outperformed irreversible ones when frozen by up to 5mAP on COCO. This is a promising result that suggests that reversible models may be able to learn more general representations that can be used for downstream tasks, but needs more investigation.

We hope that this work will help spur further research into reversible models and their applications, especially as promising alternatives to the standard use of vanilla transformers.

Bibliography

- [1] Alexei Baevski et al. “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 12449–12460. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf.
- [2] Jens Behrmann et al. “Invertible residual networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 573–582.
- [3] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.
- [4] Bo Chang et al. “Reversible architectures for arbitrarily deep residual neural networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.
- [5] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. “Per-Pixel Classification is Not All You Need for Semantic Segmentation”. In: 2021.
- [6] Krzysztof Choromanski et al. “Rethinking attention with performers”. In: *arXiv preprint arXiv:2009.14794* (2020).
- [7] Laurent Dinh, David Krueger, and Yoshua Bengio. “Nice: Non-linear independent components estimation”. In: *arXiv preprint arXiv:1410.8516* (2014).
- [8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real nvp”. In: *arXiv preprint arXiv:1605.08803* (2016).
- [9] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2021. URL: <https://openreview.net/forum?id=YicbFdNTTy>.
- [10] Haoqi Fan et al. “Multiscale vision transformers”. In: *ICCV*. 2021.
- [11] Marc Finzi et al. “Invertible convolutional networks”. In: *Workshop on Invertible Neural Nets and Normalizing Flows, International Conference on Machine Learning*. 2019.

- [12] Aidan N Gomez et al. “The Reversible Residual Network: Backpropagation Without Storing Activations”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.
- [13] Aidan N Gomez et al. “The reversible residual network: Backpropagation without storing activations”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 2211–2221.
- [14] Tristan Hascoet et al. “Layer-Wise Invertibility for Extreme Memory Cost Reduction of CNN Training”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.
- [15] Jonathan Ho et al. “Flow++: Improving flow-based generative models with variational dequantization and architecture design”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2722–2730.
- [16] Justin Johnson et al. “CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning”. In: *CVPR*. 2017.
- [17] Diederik P Kingma and Prafulla Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: *arXiv preprint arXiv:1807.03039* (2018).
- [18] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. “Reformer: The efficient transformer”. In: *arXiv preprint arXiv:2001.04451* ().
- [19] Duo Li and Shang-Hua Gao. “m-RevNet: Deep Reversible Neural Networks with Momentum”. In: *arXiv preprint arXiv:2108.05862* (2021).
- [20] Yanghao Li et al. “Exploring plain vision transformer backbones for object detection”. In: *arXiv preprint arXiv:2203.16527* (2022).
- [21] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.
- [22] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).
- [23] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.
- [24] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.
- [25] Karttikeya Mangalam et al. “Reversible Vision Transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10830–10840.
- [26] Michael E Sander et al. “Momentum residual neural networks”. In: *arXiv preprint arXiv:2102.07870* (2021).

BIBLIOGRAPHY

24

- [27] Yang Song, Chenlin Meng, and Stefano Ermon. “Mintnet: Building invertible neural networks with masked convolutions”. In: *arXiv preprint arXiv:1907.07945* (2019).
- [28] Yi Tay et al. “Sparse sinkhorn attention”. In: *Proc. ICML*. PMLR, 2020, pp. 9438–9447.
- [29] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.
- [30] Sinong Wang et al. “Linformer: Self-attention with linear complexity”. In: *arXiv preprint arXiv:2006.04768* (2020).