REVBIFPN: THE FULLY REVERSIBLE BIDIRECTIONAL FEATURE PYRAMID NETWORK

Vitaliy Chiley [1] **Vithursan Thangarasa** [2] **Abhay Gupta** [2] **Anshul Samar** [2] **Joel Hestness** [2] **Dennis DeCoste** [1]

ABSTRACT

This work introduces RevSilo, the first reversible bidirectional multi-scale feature fusion module. Like other reversible methods, RevSilo eliminates the need to store hidden activations by recomputing them. However, existing reversible methods do not apply to multi-scale feature fusion and are, therefore, not applicable to a large class of networks. Bidirectional multi-scale feature fusion promotes local and global coherence and has become a de facto design principle for networks targeting spatially sensitive tasks, e.g., HRNet (Sun et al., 2019a) and EfficientDet (Tan et al., 2020). These networks achieve state-of-the-art results across various computer vision tasks when paired with high-resolution inputs. However, training them requires substantial accelerator memory for saving large, multi-resolution activations. These memory requirements inherently cap the size of neural networks, limiting improvements that come from scale. Operating across resolution scales, RevSilo alleviates these issues. Stacking RevSilos, we create RevBiFPN, a fully reversible bidirectional feature pyramid network. RevBiFPN is competitive with networks such as EfficientNet while using up to 19.8x lesser training memory for image classification. When fine-tuned on MS COCO, RevBiFPN provides up to a 2.5% boost in AP over HRNet using fewer MACs and a 2.4x reduction in training-time memory.

1 INTRODUCTION

State-of-the-art (SOTA) computer vision (CV) networks have large memory requirements that complicate training and limit scalability. Tan & Le (2019) and Dollár et al. (2021) show how compound scaling, i.e., scaling input resolution, network width, and depth, results in efficient networks across a wide range of parameters and MAC (multiply-accumulate) counts. Even when resources are optimally allocated, scaling networks produce large feature maps. Thus, training requires a large amount of accelerator memory (Figure 1). While low-resolution intermediate representations work well for classification tasks (LeCun et al., 1998; Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; Tan & Le, 2019), dense prediction tasks, such as detection and segmentation, require the construction of spatially informative, high-resolution feature maps which further exacerbates memory issues.

U-Net (Ronneberger et al., 2015), used for segmentation, was one of the first multi-scale feature fusion networks. Initially, detection networks would perform multi-scale inference by processing every scale of an image pyramid independently. But, they soon adopted multi-scale feature fusion to directly produce a feature pyramid, i.e., multi-scale features (Lin et al., 2017a;b; Redmon & Farhadi, 2018). Bidirectional multi-scale feature fusion networks iteratively merge information between high and low-resolution feature maps, producing robust (Hendrycks & Dietterich, 2019) scale-invariant models. These models promote local and global coherence by iteratively aligning the semantic representations of fine-grained and high-level features. As a result, these networks are often the backbone of SOTA computer vision systems (Liu et al., 2018; Cai & Vascon-

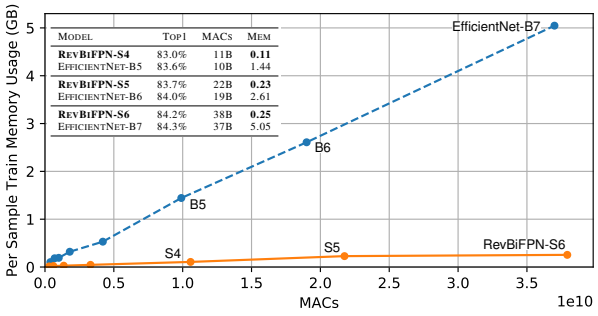

Figure 1. **MACs vs. Measured Memory Usage for ImageNet Training**: RevBiFPN significantly outperforms EfficientNet at all scales. In particular, RevBiFPN-S6 achieves comparable accuracy (84.2%) to EfficientNet-B7 on ImageNet while using comparable MACs (38.1B) and 19.8x lesser training memory per sample. Details in Tables 4 and 12.

[1]Work done while at Cerebras Systems Inc [2]Cerebras Systems Inc, California, USA. Correspondence to: Vitaliy Chiley <vitaliy@mosaicml.com>, Vithursan Thangarasa <vithu@cerebras.net>, Abhay Gupta <abhay@cerebras.net>.

Proceedings of the 6^{th} MLSys Conference, Miami Beach, FL, USA, 2023. Copyright 2023 by the author(s).

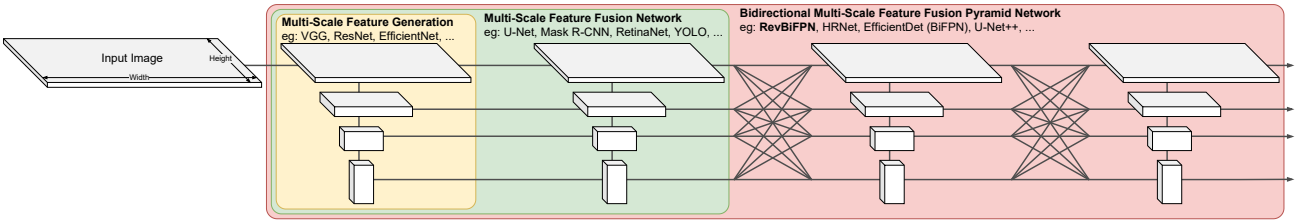

Figure 2. **Connectivity of multi-scale networks**: Features are depicted as boxes, and the lines represent the possible connectivity of networks processing features at multiple scales. Networks like VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), and EfficientNet (Tan & Le, 2019) can generate multi-scale features (yellow box). These features are often fused by networks such as U-Net (Ronneberger et al., 2015), Mask R-CNN (He et al., 2017), or YOLO (Redmon & Farhadi, 2018) for completing spatially sensitive tasks (green box). Low-resolution features communicate global information, while high-resolution features capture detailed local features such as texture and object boundaries. By iteratively mixing these features, bidirectional multi-scale feature fusion networks such as HRNet (Sun et al., 2019a), EfficientDet (Tan et al., 2020), and UNet++ (Zhou et al., 2018) promote local and global coherence, boosting performance (red box). See Appendix A for more details.

celos, 2018; Sun et al., 2019a; Tan et al., 2020). But, the memory requirements of backpropagating through multi-scale feature fusion complicate training and limit scalability. Training CV networks push the memory bounds of modern hardware, with hardware memory setting a hard limit on how far researchers scale these networks, enforcing an upper bound on network performance.

Hidden activations are needed to compute the gradient of the loss with respect to a neural network's parameters. Traditionally, the activations computed during the forward pass are cached for use in the backward pass. While this method of neural network training has worked well in the past, the growth of neural networks has outpaced increases in accelerator memory. Motivated by flow structures (Dinh et al., 2017; Kingma & Dhariwal, 2018), Gomez et al. (2017) recognized that if a network is designed using a series of invertible operations, the activations can be recomputed during the backward pass. Using this paradigm, reversible networks perform "backpropagation without storing activations" (Gomez et al., 2017), reducing their activation memory complexity with respect to depth from linear to constant. Although reversible structures have been successfully used in image classification (Gomez et al., 2017; Jacobsen et al., 2018) and language modeling (Kitaev et al., 2020; MacKay et al., 2018), they have yet to be used where they are needed most: in multi-scale feature fusion networks to produce high-resolution feature maps.

1.1 Contributions

To address the memory challenges of training models for spatially sensitive tasks, this work introduces the RevSilo and the network built with it, RevBiFPN. The main contributions of this work are:

1. The RevSilo (Figure 3), the first bidirectional multi-scale feature fusion module that is invertible.
2. RevBiFPN (Figure 4) is the first fully reversible bidirectional multi-scale feature fusion pyramid network. It is built using the RevSilo and uses a fraction of the memory compared to the same network without reversible recomputation (Figures 5 and 6).
3. With a classification head, RevBiFPN is pretrained on ImageNet (Deng et al., 2009) to accuracies competitive with networks designed specifically for classification (Figure 1 and Section 5.1).
4. To our knowledge, this work is the first to fine-tune a reversible backbone on downstream CV tasks. With the appropriate heads, RevBiFPN is competitive with similar networks on detection and segmentation tasks while using a fraction of the accelerator memory for training (Section 5.2).

The reference implementation and model checkpoints for ImageNet are available at `https://github.com/CerebrasResearch/RevBiFPN`.

2 Background

Systems using low-resolution features were often applied to image pyramids for detection (Girshick, 2015; Ren et al., 2015; Redmon et al., 2016; Redmon & Farhadi, 2017). Lin et al. (2017a) augment a pretrained classification network with a low-resolution to a high-resolution decoder to perform multi-scale feature fusion similar to the U-Net design. Rather than a single high-resolution feature map, the network outputs features from multiple spatial resolutions to create a feature pyramid. The success of the Feature Pyramid Network (FPN) motivated similar methodologies to be used throughout the computer vision community (Redmon & Farhadi, 2018; Bochkovskiy et al., 2020; He et al., 2017; Lin et al., 2017b; Goyal et al., 2021). Bidirectional multi-scale feature fusion pyramid networks (BiFPNs) further improve performance by iteratively applying multi-scale feature fusion modules (Tan et al., 2020; Ghiasi et al., 2019; Liu et al., 2018; Cai & Vasconcelos, 2018; Chen et al.,

Table 1. Memory and computational complexity of different memory-saving methods with respect to the depth of the network (D). When training in layer pipeline mode (Pétrowski et al., 1993; Kosson et al., 2021), activation complexity is quadratic with respect to depth. While gradient checkpointing decreases activation memory complexity from $O(D^2)$ to $O(D^{1.5})$ (Yang et al., 2021), reversible recomputation decreases it to $O(D)$. Both methods have a $O(D)$ overhead in the backward pass to re-materialize or recompute the activations.

	MEMORY		COMPUTE	
	LAYER SEQUENTIAL	PIPELINED PARALLEL	FORWARD PASS	BACKWARD PASS
SGD BASELINE	$O(D)$	$O(D^2)$	$O(D)$	$O(2D)$
WITH CHECKPOINTING	$O(\sqrt{D})$	$O(D^{\frac{3}{2}})$	$O(2D)$	$O(2D)$
WITH REVERSBLE RECOMPUTATION	$O(1)$	$O(D)$	$O(2D)$	$O(2D)$

2018b). This allows local information from high-resolution feature maps to be repeatedly fused with global information from low-resolution feature maps (Figure 2).

FPNs are often created using feature fusion modules to augment existing classification networks that are not designed for feature fusion. However, Zhou et al. (2015), Jacobsen et al. (2017), Ke et al. (2017), Huang et al. (2018b), Sun et al. (2019a), Sun et al. (2019b), Wang et al. (2020), Cheng et al. (2020), Fan et al. (2021), and Li et al. (2021) advocate for treating bidirectional multi-scale feature fusion as a first-class design principle in computer vision networks and show the effectiveness of this approach for classification, detection, and segmentation. Bidirectional multi-scale feature fusion networks reduce the semantic gap between consecutive feature maps (Zhou et al., 2018). By outputting a feature pyramid, these networks are also scale invariant. This improves performance, but their memory requirements complicate training and limit scalability.

Achieving SOTA results frequently requires bidirectional multi-scale feature fusion pyramid networks, or BiFPN style networks, to process mega-pixel images. This can result in a single sample's activations consuming all accelerator memory (Tao et al., 2020). Distributed training setups can accelerate these workloads but impose other limitations. For instance, using small batch sizes precludes using Batch Normalization (Ioffe & Szegedy, 2015), requiring different normalization methods (Wu & He, 2018; Chiley et al., 2019; Rao & Sohl-Dickstein, 2020; Labatie et al., 2021). Alternatively, researchers can adopt model parallel approaches to scaling models, but this often results in hardware utilization or network optimization issues (Huang et al., 2019; Chen et al., 2018a; Narayanan et al., 2019; Kosson et al., 2021). Another way to alleviate accelerator memory usage is to offload activations to host (Rajbhandari et al., 2021). However, for bandwidth-constrained systems, this results in poor FLOP utilization. When performing operations with low arithmetic intensity, such as non-linearities or depthwise convolutions (Lu et al., 2021; Qin et al., 2018), limited device bandwidth memory already results in poor FLOP utilization. Offloading activations to host uses bandwidth which is even further constrained, exacerbating the issue.

Alternatively, Volin & Ostrovskii (1985), Griewank & Walther (2000), Zweig (2000), Lewis (2003), Dauvergne & Hascoët (2006), Griewank & Walther (2008), Gruslys et al. (2016), Chen et al. (2016), Jain et al. (2020), and Feng & Huang (2021) propose gradient (or reverse) checkpointing where a subset of activations are recomputed instead of being stored. Network activations are needed to compute parameter gradients. Storing them produces an activation memory complexity that is linear in network depth. Checkpointing can reduce this complexity from $O(D)$ to $O(\sqrt{D})$ (Chen et al., 2016).

Reversible models (Gomez et al., 2017; MacKay et al., 2018; Brügger et al., 2019; Pendse et al., 2020; Yamazaki et al., 2021; Sander et al., 2021; Chun et al., 2020; Kitaev et al., 2020; Nestler & Gill, 2021) save memory by recomputing activations instead of storing them. This decreases the activation memory complexity from linear to constant. Reversible recomputation enables SOTA research without needing hardware with the latest memory capacity, which prolongs the useful life of existing hardware. As a result, less e-waste is produced, but it comes at the cost of recomputing activations, contributing to the carbon footprint of training reversible models. Table 1 shows the theoretical compute and memory complexity of training neural networks with different memory-saving techniques. Section 5.1 shows the effect this has in practice.

It should be noted that reversible networks relying on Reversible Residual Block (RevBlock) (Gomez et al., 2017) are not fully reversible. RevBlock cannot operate across different dimensionalities. Therefore RevNet (Gomez et al., 2017) and other networks built using RevBlocks, must cache activations in computational blocks that change tensor shape. Fully reversible models have the added benefit of being used for generation with Normalizing Flows (Dinh et al., 2014; Germain et al., 2015; Dinh et al., 2017; Kingma et al., 2016; Papamakarios et al., 2017; Kingma & Dhariwal, 2018; Huang et al., 2018a; Jacobsen et al., 2018; Keller et al., 2021) but are often not as efficient. For instance, the injective variant of i-RevNet (Jacobsen et al., 2018) is a fully reversible variant of RevNet but requires a 7x increase in size to match RevNet's performance. Other approaches to

reversible recomputation impose architectural limits (Bai et al., 2019), limit optimization (Behrmann et al., 2019; Thangarasa et al., 2019), or are computationally expensive (Behrmann et al., 2019). While any reversible model or method could be used for saving activation memory, none were previously applicable to bidirectional multi-scale feature fusion.

As existing reversible structures keep tensor dimensionality constant, they cannot be directly applied to multi-scale networks such as EfficientDet. One approach to producing high-resolution feature maps would be to apply the reversible residual block (Gomez et al., 2017) to an entire subnetwork, such as each hourglass of the Stacked Hourglass Network (Newell et al., 2016). While feasible, the entire subnetwork of activations would still need to be stored, limiting memory savings (Appendix B.1). The specific case of the hourglass design also produces high MAC count networks (Appendix B.2) and does not provide bidirectional multi-scale feature fusion with a feature pyramid output.

3 Reversible Residual Silo

The Reversible Residual Silo, or RevSilo, generalizes both affine coupling (Dinh et al., 2014) and the reversible residual block (Gomez et al., 2017) to create an invertible module for bidirectional multi-scale feature fusion. Figure 3 shows the two halves of the RevSilo with $N = 4$ spatial resolutions.

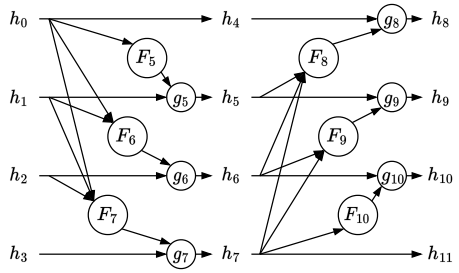

Figure 3. **An $N = 4$ RevSilo**. F_j can be any transformation; g_j can be any potentially parameterized, invertible transformation.

The left half communicates information down the feature pyramid, and the right half sends information up the feature pyramid. g_j can be any potentially parameterized, invertible transformation. In this work, g_j is element-wise addition, and therefore its inverse, g_j^{-1}, is element-wise subtraction for all j (Appendix C). If h_i and h_j are on the same row, i.e., $i \,\%\, N == j \,\%\, N$, the RevSilo's residual structure requires that the shape of h_i equals the shape of h_j. Otherwise, F_j should transform the shape of its inputs to match the shape of h_j. Besides this shape constraint, F_j can be any transformation. The RevSilo construct remains invertible even if some inputs are 0. Setting h_3 to 0 can, for example, be used to expand an $N = 3$ feature pyramid into an $N = 4$ feature pyramid. The equations for the $N = 4$ RevSilo are:

$$h_4 = h_0 \tag{1}$$
$$h_5 = g_5\left(h_1, F_5(h_0)\right) \tag{2}$$
$$h_6 = g_6\left(h_2, F_6(h_1, h_0)\right) \tag{3}$$
$$h_7 = g_7\left(h_3, F_7(h_2, h_1, h_0)\right) \tag{4}$$

followed by:

$$h_8 = g_8\left(h_4, F_8(h_7, h_6, h_5)\right) \tag{5}$$
$$h_9 = g_9\left(h_5, F_9(h_7, h_6)\right) \tag{6}$$
$$h_{10} = g_{10}\left(h_6, F_{10}(h_7)\right) \tag{7}$$
$$h_{11} = h_7 \tag{8}$$

The corresponding inverse equations are:

$$h_7 = h_{11} \tag{9}$$
$$h_6 = g_{10}^{-1}\left(h_{10}, F_{10}(h_7)\right) \tag{10}$$
$$h_5 = g_9^{-1}\left(h_9, F_9(h_7, h_6)\right) \tag{11}$$
$$h_4 = g_8^{-1}\left(h_8, F_8(h_7, h_6, h_5)\right) \tag{12}$$

followed by:

$$h_0 = h_4 \tag{13}$$
$$h_1 = g_5^{-1}\left(h_5, F_5(h_0)\right) \tag{14}$$
$$h_2 = g_6^{-1}\left(h_6, F_6(h_1, h_0)\right) \tag{15}$$
$$h_3 = g_7^{-1}\left(h_7, F_7(h_2, h_1, h_0)\right) \tag{16}$$

For the $N = 4$ RevSilo, Equations (1) to (8) are used to compute the forward pass. Instead of storing activations, they can be recomputed during the backward pass using Equations (9) to (16). While the inverse equations must be computed in order, the forward equations allow the N hidden tensors of the RevSilo to be computed simultaneously. This enables more parallelism in the resulting inference network.

It should also be noted that if, for all i, h_i is a scalar, g_j is addition, and F_j is the dot product operation for all j; then the forward equations can be rewritten as matrix-vector products with unitriangular matrices. The underlying structure that makes unitriangular matrices invertible (Thoma, 2013), makes all coupling structures (Kingma et al., 2016; Germain et al., 2015; Papamakarios et al., 2017; Huang et al., 2018a; Dinh et al., 2014; Gomez et al., 2017) invertible.

4 RevBiFPN

Reversible Bi-directional Feature Pyramid Network, or RevBiFPN, uses the RevSilo to create a fully reversible backbone that utilizes bidirectional multi-scale feature fusion and produces a feature pyramid output. Using a reversible multi-scale feature fusion module, RevBiFPN circumvents the

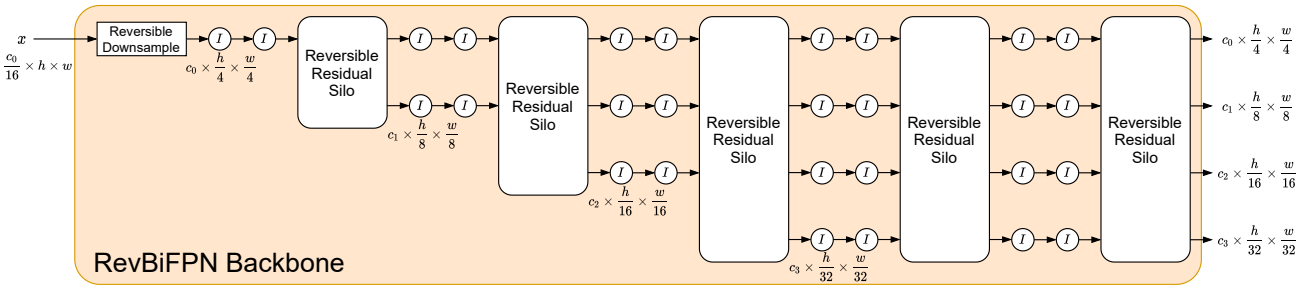

Figure 4. **A RevBiFPN that creates an $N = 4$ feature pyramid**. Given the output feature pyramid, all activations can be recomputed going backward through the network. The I components are reversible residual blocks. The network builds an $N = 4$ multi-scale hidden representation using 3 RevSilos and has an extra depth of $d = 2$ RevSilos for further feature fusion.

issues seen in the RevNet and i-RevNet design (Section 2). The high-level network structure of RevBiFPN is shown in Figure 4. The output feature pyramid can then be used as an input to different task-specific heads (Section 4.2).

The network uses the invertible SpaceToDepth stem (Ridnik et al., 2021; Shi et al., 2016; Dinh et al., 2017; Jacobsen et al., 2018) to initially downsample the input by a factor of 4 and produce $c = 4^2 \times 3 = 48$ channels. The baseline model (RevBiFPN-S0) uses $c_0 = 48$, $c_1 = 64$, $c_2 = 80$, and $c_3 = 160$ channels in its $N = 4$ spatial resolutions. As the network size increases, the input image channels are duplicated to ensure the network is fully reversible regardless of network width. The rest of the network has a structure similar to HRNet (Sun et al., 2019a) where transformations in the same spatial resolution use Reversible Residual Blocks (Gomez et al., 2017).

For simplicity, the network uses the RevSilo variant shown in Figure 19 (Appendix C). Here, the F operations independently transform and sum each input. The network isn't designed with a specific hardware target in mind and therefore uses the MBConv block (Howard et al., 2017), a building block that efficiently utilizes parameters and MACs (multiply-accumulates). [1] The MBConv block is used for both transformations in the reversible residual block and the F transformations of the RevSilo. Using the MBConv block in network design produces networks with fast inference speed on inference devices (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019; Tan & Le, 2019; Mehta & Rastegari, 2021).

Within its RevSilos, RevBiFPN upsamples and downsamples features by factors of 2. To upsample a feature by a factor of 2^k, the depthwise convolution of the MBConv block uses a stride of 1 and a kernel size of 3 or 5; this is then followed by bilinear upsampling. To downsample a feature by a factor of 2^k, the depthwise convolution of the MBConv block uses a stride of 2^k and a kernel size of $2^{k+1} \pm 1$. As a result, the network uses a diverse set of kernel sizes as suggested by Tan et al. (2019). Network parameters are initialized using Kaiming Initialization (He et al., 2015). Batch Normalization biases are initialized to zero, and weights are initialized to one, except the weights of the last normalization layer, which are initialized to zero to promote stability (Kingma & Dhariwal, 2018).

The network uses the MBConv variant with squeeze-excite layers (Tan & Le, 2019) and the hard-swish non-linearity (Howard et al., 2019). The network has larger expansion ratios on the lower resolution streams and uses larger squeeze-excite ratios on the large resolution streams (Ridnik et al., 2021). With a classification head, the resulting network has a parameter and MAC profile similar to common classification networks. RevBiFPN-S0 is then scaled (Section 4.3) and compared to other network families on the commonly used ImageNet (Deng et al., 2009) benchmark. The RevBiFPN family of networks is pretrained on ImageNet and fine-tuned with task-specific heads (Section 4.2) for object detection and instance segmentation on MS COCO (Lin et al., 2014).

4.1 Memory Savings

The activation memory complexity of training a CV network is $\mathrm{O}(nchwd)$ where n is the batch size, c is the number of channels representing the network's width, h and w specify the input resolution, and d denotes the depth of the network. By decoupling depth from activation memory requirements, reversible networks have an activation memory complexity of $\mathrm{O}(nchw)$. Figure 5 shows the measured memory usage of the RevBiFPN-S0 network as the network depth is scaled with and without reversible recomputation. This demonstrates that measured memory usage is approximately constant when reversible recomputation is used but increases linearly otherwise.

When scaling width, batch size, or input resolution, networks with and without reversible recomputation have the same complexity, but using reversibility creates a mem-

[1] Many research papers often report MACs as FLOPs, which is incorrect. This work uses MAC to mean multiply-accumulate, as FLOP is a single floating point operation.

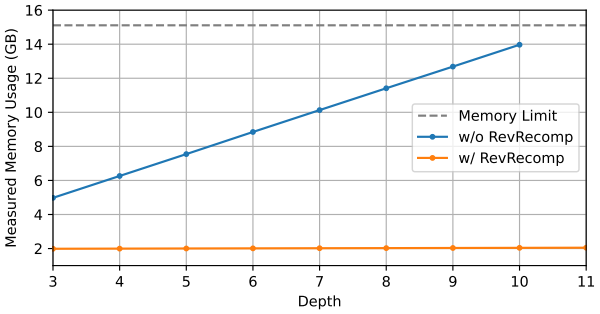

Figure 5. Single GPU memory usage with batch size 64 for training RevBiFPN-S0 on ImageNet with and without reversible recomputation (RevRecomp) as depth is scaled. Reversible recomputation decreases the activation memory complexity from linear to constant. These memory savings can be reallocated to scaling network width and input resolution to produce RevBiFPN S1-S6.

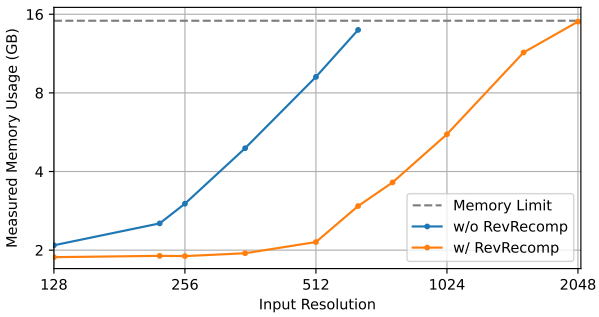

Figure 6. The measured activation memory of training a network using a batch size of 16 on a single GPU with and without reversible recomputation (RevRecomp) as the input resolution is scaled. We observe that with RevRecomp, we can train with higher input resolutions, which is useful in domains such as object detection and segmentation.

ory offset that enables larger variants to be trained. As an example, Figure 6 shows the measured memory usage of RevBiFPN-S0 as the resolution is varied. The reversible variant has an advantageous offset and can run resolutions about 4^2 larger than is possible with a network without reversible recomputation. On a 16GB system, the largest image a network without reversible recomputation can process is just over 2K×2K. With reversibility, the same network can process images with resolutions up to 8K×8K.

4.2 Network Heads

While the RevBiFPN backbone is fully reversible, it can be used with non-reversible heads. Before each head is applied, a set of MBConv blocks is used as a neck, with reverse checkpointing, to transform the output channels of RevBiFPN-S0 to 48, 64, 128, and 320. The dimensionality of the neck and heads is scaled using the width multipliers shown in Table 2 (Section 4.3). For the detection and segmentation networks, the input resolution is also modified.

ImageNet classification is used to pretrain the RevBiFPN backbone before it is fine-tuned for object detection and segmentation. The backbone outputs a feature pyramid transformed by the neck and non-reversible classification head shown in Figure 7. In the head, the highest resolution feature map is downsampled by a factor of 2 using an MBConv block with stride 2 and is added to the next largest feature map. This is repeated multiple times until all information is aggregated into the lowest-resolution feature map. A 1×1 convolution is applied, followed by global average pooling and a dense layer. This design is inspired by Sun et al. (2019a) but uses the MBConv block.

Object detection and instance segmentation are done with the Faster R-CNN and Mask R-CNN heads provided in MMDetection (Chen et al., 2019).

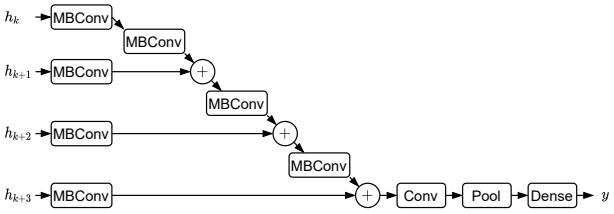

Figure 7. Neck and classification head with feature pyramid input.

Table 2. Network width multiplier (m_w), depth (d), and input height and width (h and w) of RevBiFPN variants trained on ImageNet at different scales. Without reversibility, the training setup must be modified to accommodate scales past RevBiFPN-S1. Reversibility enables training RevBiFPN-S6 with an activations that are about 24x larger than that of RevBiFPN-S1.[3]

MODEL	m_w	d	h AND w
REVBIFPN-S0	1	2	224
REVBIFPN-S1	1.3	2	256
REVBIFPN-S2	2	2	256
REVBIFPN-S3	2.7	3	288
REVBIFPN-S4	4	4	320
REVBIFPN-S5	5.3	4	352
REVBIFPN-S6	6.7	5	352

4.3 Network Scaling

Once the baseline network is designed, scaling the input resolution, width, and depth generally results in better performance. Classically, networks such as VGG (Simonyan & Zisserman, 2015) and ResNet (He et al., 2016) focus on scaling network depth. Tan & Le (2019) shows how compound scaling, i.e., scaling all dimensions, results in

[3]Without reversible recomputation, the memory used for activations, $n \times c \times h \times w \times d$, dominates memory usage. $|\text{RevBiFPN-S6}| \,/\, |\text{RevBiFPN-S1}| = (n \times 6.67c \times 352 \times 352 \times 5)/(n \times 1.33c \times 256 \times 256 \times 2) = 23.7$.

Table 3. Single-crop, single-model ImageNet accuracy.

MODEL	PARAMS	MACS	TOP1
REVBIFPN-S0	3.42M	0.31B	72.8%
REVBIFPN-S1	5.11M	0.62B	75.9%
REVBIFPN-S2	10.6M	1.37B	79.0%
REVBIFPN-S3	19.6M	3.33B	81.1%
REVBIFPN-S4	48.7M	10.6B	83.0%
REVBIFPN-S5	82.0M	21.8B	83.7%
REVBIFPN-S6	142.3M	38.1B	84.2%

efficient networks at all parameters and MAC counts. Dollár et al. (2021) shows how to scale such that the network run-time is minimized for large networks. Equations (4) and (5) of Dollár et al. (2021) produce a "family of scaling strategies parameterized by α."

This work uses these scaling strategies but sets $\alpha = 2/3$. While Dollár et al. (2021) recommends $\alpha = 4/5$, they also show $\alpha = 2/3$ is nearly as fast but prioritizes depth and resolution scaling. This gives added memory benefits in the reversible setting (Section 4.1). Given the outputs of the scaling strategy, m_w is chosen such that channel counts are multiples of 16, the depth is rounded to the nearest integer, and the resolution is set to a multiple of 2^5 (Table 2).

5 EXPERIMENTS

5.1 ImageNet Classification

Setup. The ImageNet dataset is used to pretrain the network before variants are fine-tuned on downstream tasks. All RevBiFPN variants are pretrained for 350 epochs using 8 GPUs with a per GPU batch size of 64. This enables higher throughput for training while amortizing the costs of reverisble recomputation. SGD is used with a learning rate of 0.1 and momentum of 0.9, and an exponential moving average (EMA) of the network parameters is used with a decay of 0.9999. A 5 epoch learning rate warm-up is used with a starting learning rate of 10^{-3} followed by cosine decay (Loshchilov & Hutter, 2017). The last ten epochs use a constant learning rate of 10^{-4}. The network uses batch normalization with a momentum of 0.9 and epsilon of 10^{-3}. Training is regularized using label smoothing (Szegedy et al., 2016), weight decay, dropout (Srivastava et al., 2014), stochastic depth (Huang et al., 2016), CutMix (Yun et al., 2019), mixup (Zhang et al., 2018), and RandAugment (Cubuk et al., 2020). Tuning these parameters could result in further improvement (details in Appendix E).

Results. Although RevBiFPN-S0 and RevBiFPN-S1 can be trained without reversible recomputation, unless otherwise stated, all of the results are shown for networks trained with reversible recomputation. Table 3 summarizes ImageNet classification results showing top1 accuracy, parameter counts and MACs used at evaluation. While not designed primarily for classification, RevBiFPN still produces results comparable to classification-specific networks. RevBiFPN-S6 uses 38.1B MACs and achieves 84.2% ImageNet Top1 accuracy, making it comparable to EfficientNet-B7, which uses 37B MACs to achieve 84.3% ImageNet Top1 accuracy. Table 12 in the appendix extends Table 3 to include comparisons with other networks.

Table 4. Training Memory (GB) used per sample. The training resolution used for RevBiFPN-S6 is 352 and EfficientNet-B7 is trained with a resolution of 600.

MODEL	INPUT RESOLUTION		
	TRAIN RES	224	384
REVBIFPN-S6	**0.254**	**0.086**	**0.291**
EFFICIENTNET-B7	5.047	0.673	1.786

Table 5. Slowdown of using reversible recomputation. Theoretically reversible recomputation adds a 33% compute overhead (Table 1). In practice, with more memory, training can be optimized and can use a larger batch size to help efficiency.

MODEL	SLOWDOWN
REVBIFPN-S0	25.02%
REVBIFPN-S2	21.96%
REVBIFPN-S4	15.73%
REVBIFPN-S6	12.73%

Table 4 shows that the GPU memory usage of RevBiFPN-S6 is a fraction of the memory used by EfficientNet-B7 at their respective training resolutions and at input resolutions of 224 and 384, which are frequently used in CV backbones.

Table 5 shows the measured slowdowns from using reversible recomputation for different networks. We see that as the networks get larger, the overheads reduce considerably, making the technique efficient to use at scale.

5.1.1 *Training With Reversibility*

Under infinite precision, training with and without reversible recomputation would produce identical results. Figure 8 shows that while there are differences when using finite precision, these are inconsequential. Training RevBiFPN-S0 with reversible recomputation requires only 2GB of accelerator memory and produces results nearly indistinguishable from regular training, which consumes 12GB of memory.

5.1.2 *Architectural Ablations*

As noted in Section 4, RevBiFPN is structurally similar to HRNet. In this section, RevBiFPN is trained on ImageNet for 150 epochs at an input resolution of 96×96 to ablate architectural design decisions.

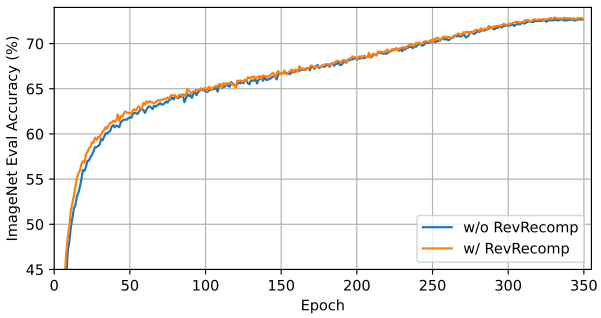

Figure 8. ImageNet validation accuracy when training RevBiFPN-S0 with and without reversible recomputation (RevRecomp).

Table 6. Down & Up Sampling Operation's influence on Accuracy.

DOWN / UP SAMPLING	PARAMS	MACS	TOP1
LD / SU	3.49M	75.7M	61.5%
SD / SU	3.28M	67.2M	60.8%
SD / LU	3.47M	69.5M	61.5%

Down and Up Sampling Operation. HRNet uses k stride 2 convolution blocks to downsample by 2^k (LD). An alternative downsampling schema would use a single block with stride 2^k and an increased kernel size such that the entire input is used to produce the output (SD). HRNet uses a 1×1 convolution paired with an upsample operation in the 'nearest' mode (SU) to upsample feature maps. The 1×1 convolution does not operate in the spatial domain, and the upsample operation is in the 'nearest' mode. This is ablated by changing the upsampling block to use a 3×3 convolution paired with an upsample operation in 'bilinear' mode (LU).

While replacing LD with SD curbs accuracy on ImageNet, augmenting this change by replacing SU with LU results in a total MAC decrease of about 8% while not affecting ImageNet accuracy (Table 6).

Backbone Stem. Common practice dictates using a convolutional stem for neural network design. Ridnik et al. (2021) proposes replacing this with the SpaceToDepth stem. Their work shows this does not affect network accuracy while increasing GPU throughput performance. Table 7 reaffirms their results and highlights the resulting MAC decrease.

Squeeze-Excite. Ridnik et al. (2021) notes that when applied to "low-resolution maps, Squeeze-Excite does not get a large accuracy benefit from the global average pooling operation that SE provides." They advocate for using Squeeze-Excite on large spatial resolutions as opposed to small spatial resolutions, as this provides a good accuracy vs. throughput tradeoff. Table 8 affirms their result by showing how Squeeze-Excite, when applied to the low-resolution path, leaves accuracy relatively unaffected, but when applied to the high-resolution path, improves performance.

Table 7. Stem's influence on Accuracy.

STEM	PARAMS	MACS	TOP1
CONVOLUTIONAL	3.49M	75.7M	61.5%
SPACETODEPTH	3.49M	73.7M	61.5%

Table 8. Influence of Squeeze-Excite.

SQUEEZE-EXCITE	PARAMS	MACS	TOP1
NONE	3.40M	75.5M	61.3%
LOW-RES PATH	3.49M	75.7M	61.4%
HIGH-RES PATH	3.46M	76.1M	61.6%

5.2 MSCOCO Object Detection and Instance Segmentation

Setup. Experimental results are presented on the MS COCO 2017 detection dataset, which contains about 118k images for training and 5k for validation (`minival`). The average precision (AP) metric is adopted, which is the standard COCO evaluation procedure. The multi-level feature representations from RevBiFPN, as shown in Figure 4, are applied for object detection. There is no additional data augmentation besides standard horizontal flipping. For training and testing, the input images are resized so that the shorter edge is 800 pixels (Lin et al., 2017a). Evaluation is performed using a single image scale.

RevBiFPN is compared with HRNet and ResNet. The object detection performance is evaluated on COCO `minival` under the two-stage anchor-based framework, Faster-RCNN (Ren et al., 2015). Faster R-CNN models are trained using RevBiFPN, HRNet, and ResNet-FPN as pretrained backbones on the MMDetection open-source object detection toolbox (Chen et al., 2019) using the provided training configurations. Figures 9 to 12 summarizes the results. Tables 13 and 14 in Appendix G extend these results to show parameters, the evaluation MACs per sample, the GPU memory usage during training, and scores. Similar to Sun et al. (2019a), the GPU memory usage is measured during training on a 4 GPU system, with an input size of 800×1333 and batch size of 8.

Baseline results are from Wang et al. (2020). RevBiFPN is pretrained for 350 epochs whereas Wang et al. (2020) pretrains for 100 epochs. He et al. (2019) shows how longer fine-tuning schedules can eliminate the benefits of pretraining. Although not ideal, this provides a way to compare networks trained for 100 epochs and fine-tuned with a 2x schedule to networks trained for 350 epochs and fine-tuned with a 1x schedule. Even though most works would only compare networks fine-tuned with the same schedules, 1x and 2x schedules are included to enable such comparisons. Tan et al. (2020) fine-tunes networks for up to 600 epochs.

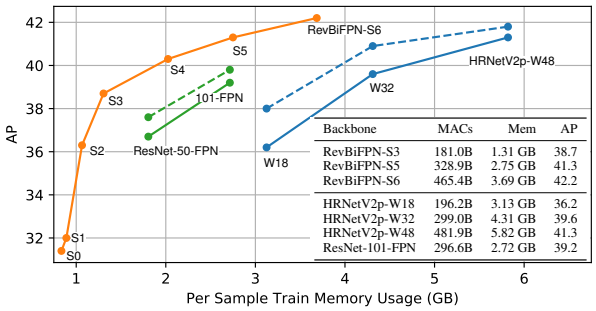

Figure 9. Object detection results on COCO `minival` in the Faster R-CNN framework as a function of memory used for training. Tables show results for 1x schedule.

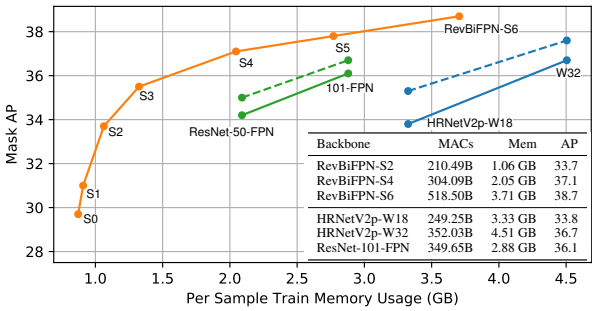

Figure 10. Instance segmentation results on COCO `minival` in the Mask R-CNN framework as a function of memory used for training. Tables show results for 1x schedule.

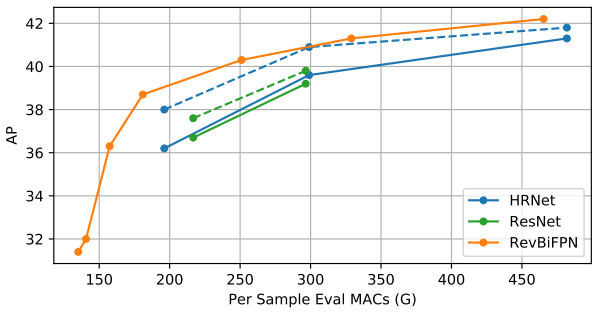

Figure 11. Object detection results on COCO `minival` in the Faster R-CNN framework as a function of evaluation MACs.

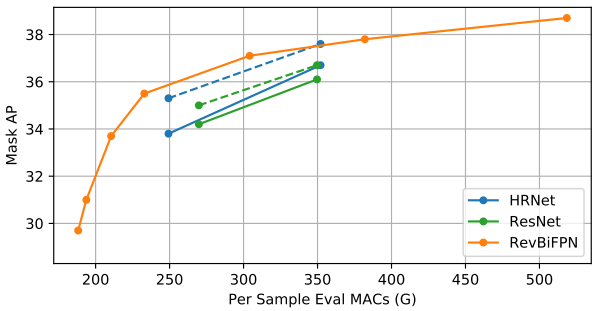

Figure 12. Instance segmentation results on COCO `minival` in the Mask R-CNN framework as a function of evaluation MACs.

As a result, EfficientDet (Tan et al., 2020) serves as a strong baseline for work pursuing SOTA results. Being subject to resource constraints, we do not make such comparisons and instead focus on memory saving. However, note that Tan et al. (2020) shows how longer training schedules further differentiate networks using bidirectional multi-scale feature fusion.

Results. Figures 9 to 12 show how RevBiFPN, with a 1x fine-tuning schedule, uses less memory and compute[4] than the baseline networks at different network performance levels even when those networks are fine-tuned using a 2x schedule. In Figures 9 to 14, networks fine-tuning using a 1x are shown using solid lines, networks fine-tuning using a 2x are shown using dashed lines. RevBiFPN is fine-tuned using the HRNet training configurations. Tuning the hyperparameters could further improve these results.

The Faster R-CNN (He et al., 2017) framework is used to evaluate RevBiFPN for object detection on MS COCO. The results are obtained on the MMDetection toolbox and are summarized in Figures 9 and 11 and are extended in Table 13. Figure 9 shows that RevBiFPN-S5 achieves an absolute gain of 3.3% in AP over HRNetV2p-W18 fine-tuned using the 2x schedule while uses 0.75GB less memory. RevBiFPN-S3 achieves an absolute gain of 2.5% in AP over HRNetV2p-W18 using fewer MACs and a ~2.4x reduction in training-time memory usage and still outperforms HRNetV2p-W18 by 0.7% AP even if it is tuned using a 2x schedule. When the scaled HRNetV2p-W48 is tuned using a 2x schedule, it uses ~1.6x the memory and still does not outperform the RevBiFPN-S6 variant tuned using the 1x schedule.

The Mask R-CNN (He et al., 2017) framework is used to evaluate RevBiFPN for object detection and instance segmentation on MS COCO. Results are obtained using the MMDetection toolbox and are summarized in Figures 10 and 12 and are extended in Table 14. The overall performance of RevBiFPN-S2 is comparable to HRNetV2p-W18 but uses ~1.2x fewer MACs and ~2.5x less GPU memory during training. RevBiFPN-S6 outperforms HRNetV2p-W32 by 2% Mask AP and 2.4% Bbox AP while using 1.6GB less memory and still outperforms HRNetV2p-W32 when it is fine-tuned using the 2x schedule. Generally, RevBiFPN enables larger batch sizes and image resolutions for detection and segmentation.

[4]The tool used to analyze the evaluation MACs per sample for the various models can be found here: https://github.com/open-mmlab/mmcv/blob/master/mmcv/cnn/utils/flops_counter.py

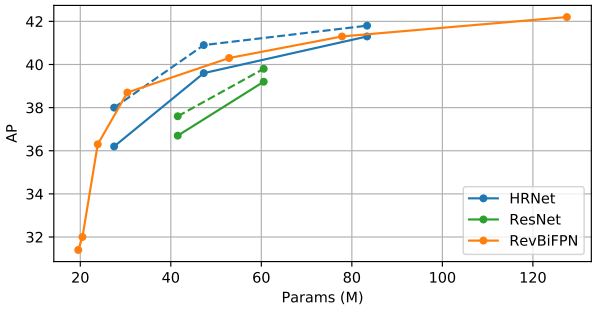

Figure 13. Object detection results on COCO `minival` in the Faster R-CNN framework as a function of network parameters.

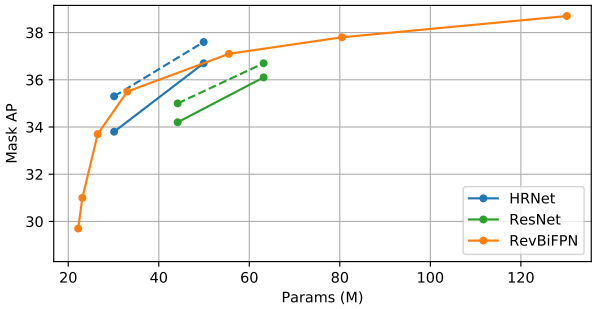

Figure 14. Instance segmentation results on COCO `minival` in the Mask R-CNN framework as a function of network parameters.

Figures 13 and 14 plot the performance of RevBiFPN as a function of network parameters, comparing it to ResNet-FPN and HRNet baselines. When fine-tuned with a 1x schedule, RevBiFPN outperforms the baseline networks. HRNet is only able to, per parameter, outperform RevBiFPN when fine-tuned with a 2x schedule (compared to RevBiFPN fine-tuned with a 1x schedule). While HRNet can be competitive when using parameter count as a metric, Dehghani et al. (2021) and Mehta & Rastegari (2021) show that network parameter counts produce misleading notions of efficiency and should generally not be used for comparing networks. The efficiency of RevBiFPN is better communicated by Figures 11 and 12 where Figures 9 and 10 shows the memory saving provided by RevBiFPN.

6 Conclusion

Bidirectional multi-scale feature fusion has driven progress in computer vision, but accelerator memory often limits network scale. Reversible methods decrease the activation memory complexity with respect to the depth from linear to constant but were previously not applicable to bidirectional multi-scale feature fusion. This work introduces RevSilo, a reversible bidirectional multi-scale feature fusion module. This enables the training of BiFPN-style networks without storing activations. The RevSilo is used to design RevBiFPN, which is competitive on classification, segmentation, and detection tasks, all while using a fraction of the memory for training. RevBiFPN applies to memory-constrained settings such as high-resolution detection and segmentation and enables SOTA research without needing hardware with the latest memory capacity.

6.1 Future Work

RevBiFPN is developed to efficiently train networks that drive semantic coherence across feature scales for tasks such as object detection and semantic segmentation. Tasks such as human pose estimation (Newell et al., 2016) and large-scale medical segmentation (Zhou et al., 2018) can also benefit from such efficiency gains. Awiszus et al. (2020) argue that multi-scale processing is needed to generate images using Generative Adversarial Networks (GANs). Prior to this work, bidirectional multi-scale feature fusion wasn't possible in flow models, but now the RevSilo and RevBiFPN enable multi-scale fusions in flow-based generation.

While we focus on developing methods and networks for efficient training, tuning models for inference efficiency is also an active direction of research (Mehta & Rastegari, 2021; Sandler et al., 2018; Ding et al., 2021). To motivate using alternate hardware accelerators, such as the Cerebras Wafer Scale Engine (Lie, 2022b;a), we use MBConv as the primary building block of the network. However, based on the target inference device, one can also swap out the basic building block to the ResNet (He et al., 2016) or Transformer block (Vaswani et al., 2017). Also, methods such as pruning (Han et al., 2015b; Yu et al., 2017) and quantization (Han et al., 2015a; Krishnamoorthi, 2018) are standard for supporting real-world inference scenarios. The impact of reversible recomputation on such methods has yet to be explored in the context of the RevSilo. Finally, while we rely on Fast Scaling (Dollár et al., 2021) for our models, researchers can use alternative strategies to scaling models such as Neural Architecture Search (Tan et al., 2019; Ghiasi et al., 2019; Tan & Le, 2019), while adding memory constraints (Zhao et al., 2021) to derive model configurations that satisfy inference requirements.

Acknowledgements

We thank Shreyas Saxena, Nolan Dey, and Valentina Popescu for their help and comments that improved the manuscript. We also thank Ben Wang, Ross Wightman, Lucas Nestler, Jan XMaster, Alexander Mattick, Atli Kosson, Abhinav Venigalla, Xin Wang, Vinay Rao, and Kenyon (Chuan-Yung) Tsai for insightful discussions.

REFERENCES

Awiszus, M., Schubert, F., and Rosenhahn, B. TOAD-GAN: Coherent Style Level Generation From a Single Example. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2020.

Bai, S., Kolter, J. Z., and Koltun, V. Deep Equilibrium Models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible Residual Networks. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2019.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*, 2020.

Brock, A., De, S., Smith, S. L., and Simonyan, K. High-Performance Large-Scale Image Recognition Without Normalization. *arXiv preprint arXiv:2102.06171*, 2021.

Brügger, R., Baumgartner, C. F., and Konukoglu, E. A Partially Reversible U-Net for Memory-Efficient Volumetric Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 429–437. Springer, 2019.

Cai, Z. and Vasconcelos, N. Cascade R-CNN: Delving into High Quality Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, 2018.

Chen, C.-C., Yang, C.-L., and Cheng, H.-Y. Efficient and Robust Parallel DNN Training Through Model Parallelism on Multi-GPU Platform. *arXiv preprint arXiv:1809.02839*, 2018a.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174*, 2016.

Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. Cascaded Pyramid Network for Multi-Person Pose Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7103–7112, 2018b.

Cheng, B., Xiao, B., Wang, J., Shi, H., Huang, T. S., and Zhang, L. HigherHRNet: Scale-Aware Representation Learning for Bottom-Up Human Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5386–5395, 2020.

Chiley, V., Sharapov, I., Kosson, A., Koster, U., Reece, R., Samaniego de la Fuente, S., Subbiah, V., and James, M. Online Normalization for Training Neural Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Chun, I. Y., Huang, Z., Lim, H., and Fessler, J. A. Momentum-Net: Fast and Convergent Iterative Neural Network for Inverse Problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. CoAtNet: Marrying Convolution and Attention for All Data Sizes. *arXiv preprint arXiv:2106.04803*, 2021.

Dauvergne, B. and Hascoët, L. The Data-Flow Equations of Checkpointing in Reverse Automatic Differentiation. In *International Conference on Computational Science*, pp. 566–573. Springer, 2006.

Dehghani, M., Arnab, A., Beyer, L., Vaswani, A., and Tay, Y. The Efficiency Misnomer. *arXiv preprint arXiv:2110.12894*, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13733–13742, 2021.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-Linear Independent Components Estimation. In *International Conference on Learning Representations Workshop*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density Estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

Dollár, P., Singh, M., and Girshick, R. Fast and Accurate Model Scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 924–932, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., and Feichtenhofer, C. Multiscale Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6824–6835, October 2021.

Feng, J. and Huang, D. Optimal gradient checkpoint search for arbitrary computation graphs. In *CVPR*, 2021.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked Autoencoder for Distribution Estimation. In *International Conference on Machine Learning*, pp. 881–889. PMLR, 2015.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

Girshick, R. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The Reversible Residual Network: Backpropagation Without Storing Activations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 2211–2221, 2017.

Goyal, A., Bochkovskiy, A., Deng, J., and Koltun, V. Nondeep Networks. *arXiv preprint arXiv:2110.07641*, 2021.

Griewank, A. and Walther, A. Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1): 19–45, 2000.

Griewank, A. and Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

Gruslys, A., Munos, R., Danihelka, I., Lanctot, M., and Graves, A. Memory-Efficient Backpropagation Through Time. *Advances in Neural Information Processing Systems*, 29, 2016.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv*, 2015a.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015b.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

He, K., Girshick, R., and Dollár, P. Rethinking ImageNet Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4918–4927, 2019.

Hendrycks, D. and Dietterich, T. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2019.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., and Adam, H. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural Autoregressive Flows. In *International Conference on Machine Learning*, pp. 2078–2087. PMLR, 2018a.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep Networks with Stochastic Depth. In *European Conference on Computer Vision*, pp. 646–661. Springer, 2016.

Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., and Weinberger, K. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations*, 2018b.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. *Advances in Neural Information Processing Systems*, 32:103–112, 2019.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pp. 448–456. PMLR, 2015.

Jacobsen, J.-H., Oyallon, E., Mallat, S., and Smeulders, A. W. Multiscale Hierarchical Convolutional Networks. In *International Conference on Machine Learning*. PMLR, 2017.

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. i-RevNet: Deep Invertible Networks. In *International Conference on Learning Representations*, 2018.

Jain, P., Jain, A., Nrusimha, A., Gholami, A., Abbeel, P., Gonzalez, J., Keutzer, K., and Stoica, I. Checkmate: Breaking the Memory Wall with Optimal Tensor Rematerialization. In Dhillon, I., Papailiopoulos, D., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 497–511, 2020.

Ke, T.-W., Maire, M., and Yu, S. X. Multigrid Neural Architectures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6665–6673, 2017.

Keller, T. A., Peters, J. W., Jaini, P., Hoogeboom, E., Forré, P., and Welling, M. Self Normalizing Flows. In *International Conference on Machine Learning*, pp. 5378–5387. PMLR, 2021.

Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved Variational Inference with Inverse Autoregressive FLOW. *Advances in Neural Information Processing Systems*, 29:4743–4751, 2016.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*, 2020.

Kosson, A., Chiley, V., Venigalla, A., Hestness, J., and Koster, U. Pipelined Backpropagation at Scale: Training Large Models without Batches. In Smola, A., Dimakis, A., and Stoica, I. (eds.), *Proceedings of Machine Learning and Systems*, volume 3, pp. 479–501, 2021.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25: 1097–1105, 2012.

Labatie, A., Masters, D., Eaton-Rosen, Z., and Luschi, C. Proxy-Normalizing Activations to Match Batch Normalization while Removing Batch Dependence. *arXiv preprint arXiv:2106.03743*, 2021.

LeCun, Y. A., Bottou, L., Bengio, Y., and Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Lewis, B. Debugging Backwards in Time. *arXiv preprint cs/0310016*, 2003.

Li, Y., Wu, C.-Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., and Feichtenhofer, C. Improved Multiscale Vision Transformers for Classification and Detection. *arXiv preprint arXiv:2112.01526*, 2021.

Lie, S. Harnessing the Power of Sparsity for Large GPT AI Models. https://www.cerebras.net/blog/harnessing-the-power-of-sparsity-for-large-gpt-ai-models, 2022a.

Lie, S. Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning : Cerebras systems. In *2022 IEEE Hot Chips 34 Symposium (HCS)*, 2022b.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017a.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017b.

Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. Path Aggregation Network for Instance Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8759–8768, 2018.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv preprint arXiv:2103.14030*, 2021.

Loshchilov, I. and Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*, 2017.

Lu, G., Zhang, W., and Wang, Z. Optimizing Depthwise Separable Convolution Operations on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

MacKay, M., Vicol, P., Ba, J., and Grosse, R. B. Reversible Recurrent Neural Networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Mehta, S. and Rastegari, M. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. *arXiv preprint arXiv:2110.02178*, 2021.

MMSegmentation Contributors. MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

Narayanan, D., Harlap, A., Phanishayee, A., Seshadri, V., Devanur, N., Granger, G., Gibbons, P., and Zaharia, M. PipeDream: Generalized Pipeline Parallelism for DNN Training. In *ACM Symposium on Operating Systems Principles (SOSP 2019)*, October 2019.

Nestler, L. and Gill, D. HomebrewNLP. https://github.com/HomebrewNLP/HomebrewNLP, 2021.

Newell, A., Yang, K., and Deng, J. Stacked Hourglass Networks for Human Pose Estimation. In *European Conference on Computer Vision*, pp. 483–499. Springer, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Pendse, M., Thangarasa, V., Chiley, V., Holmdahl, R., Hestness, J., and DeCoste, D. Memory Efficient 3D U-Net with Reversible Mobile Inverted Bottlenecks for Brain Tumor Segmentation. In *BrainLes@MICCAI*, 2020.

Pétrowski, A., Dreyfus, G., and Girault, C. Performance Analysis of a Pipelined Backpropagation Parallel Algorithm. *IEEE Transactions on Neural Networks*, 4 6:970–81, 1993.

Qin, Z., Zhang, Z., Li, D., Zhang, Y., and Peng, Y. Diagonalwise Refactorization: An Efficient Training Method for Depthwise Convolutions. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2018.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing Network Design Spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020.

Rajbhandari, S., Ruwase, O., Rasley, J., Smith, S., and He, Y. ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning. *arXiv preprint arXiv:2104.07857*, 2021.

Rao, V. and Sohl-Dickstein, J. Is Batch Norm Unique? An Empirical Investigation and Prescription to Emulate the Best Properties of Common Normalizers without Batch Dependence. *arXiv preprint arXiv:2010.10687*, 2020.

Redmon, J. and Farhadi, A. YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, 2017.

Redmon, J. and Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28, pp. 91–99. Curran Associates, Inc., 2015.

Ridnik, T., Lawen, H., Noy, A., Baruch, E. B., Sharir, G., and Friedman, I. TResNet: High Performance GPU-Dedicated Architecture. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1400–1409, 2021.

Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Sander, M. E., Ablin, P., Blondel, M., and Peyre, G. Momentum Residual Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 139:9276–9287. PMLR, 2021.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. Real-Time Single Image and Video Super-Resolution using an Efficient Sub-Pixel Convolutional Neural Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1874–1883, 2016.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Sun, K., Xiao, B., Liu, D., and Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5693–5703, 2019a.

Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., and Wang, J. High-Resolution Representations for Labeling Pixels and Regions. *arXiv preprint arXiv:1904.04514*, 2019b.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Tan, M. and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Tan, M. and Le, Q. V. EfficientNetV2: Smaller Models and Faster Training. *arXiv preprint arXiv:2104.00298*, 2021.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Tan, M., Pang, R., and Le, Q. V. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020.

Tao, A., Sapra, K., and Catanzaro, B. Hierarchical Multi-Scale Attention for Semantic Segmentation. *arXiv preprint arXiv:2005.10821*, 2020.

Thangarasa, V., Tsai, C.-Y., Taylor, G. W., and Köster, U. Reversible Fixup Networks for Memory-Efficient Training. In *NeurIPS Systems for ML (SysML) Workshop*, 2019.

Thoma, M. Solving Equations of Lower Unitriangular Matrices. https://martin-thoma.com/solving-equations-of-unipotent-lower-triangular-matrices/, 2013.

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jegou, H. Going Deeper With Image Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42, October 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Volin, Y. M. and Ostrovskii, G. M. Automatic Computation of Derivatives With the Use of The Multilevel Differentiating Technique—1. Algorithmic Basis. *Computers & Mathematics with Applications*, 11(11):1099–1114, 1985.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Wightman, R. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models, 2019.

Wu, Y. and He, K. Group Normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Yamazaki, K., Rathour, V. S., and Le, T. Invertible Residual Network with Regularization for Effective Medical Image Segmentation. *arXiv preprint arXiv:2103.09042*, 2021.

Yang, B., Zhang, J., Li, J., Ré, C., Aberger, C., and De Sa, C. PipeMare: Asynchronous Pipeline Parallel DNN Training. *Proceedings of Machine Learning and Systems*, 3, 2021.

Yu, J., Lukefahr, A., Palframan, D., Dasika, G., Das, R., and Mahlke, S. Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 548–560, 2017. doi: 10.1145/3079856.3080215.

Yuan, L., Hou, Q., Jiang, Z., Feng, J., and Yan, S. VOLO: Vision Outlooker for Visual Recognition. *arXiv preprint arXiv:2106.13112*, 2021.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*, 2018.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. Pyramid Scene Parsing Network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C. C., Lin, D., and Jia, J. PSANet: Point-wise Spatial Attention Network for Scene Parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 267–283, 2018.

Zhao, Y., Dong, L., Shen, Y., Zhang, Z., Wei, F., and Chen, W. Memory-efficient differentiable transformer architecture search. *arXiv preprint arXiv:2105.14669*, 2021.

Zhou, Y., Hu, X., and Zhang, B. Interlinked Convolutional Neural Networks for face parsing. In *International symposium on neural networks*, pp. 222–231. Springer, 2015.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. In *Deep Learning in Medical Image analysis and Multimodal Learning for Clinical Decision Support*, pp. 3–11. Springer, 2018.

Zweig, G. Exact Alpha-Beta Computation in Logarithmic Space with Application to MAP Word Graph Construction. In *Proceedings of ICSLP*, January 2000.

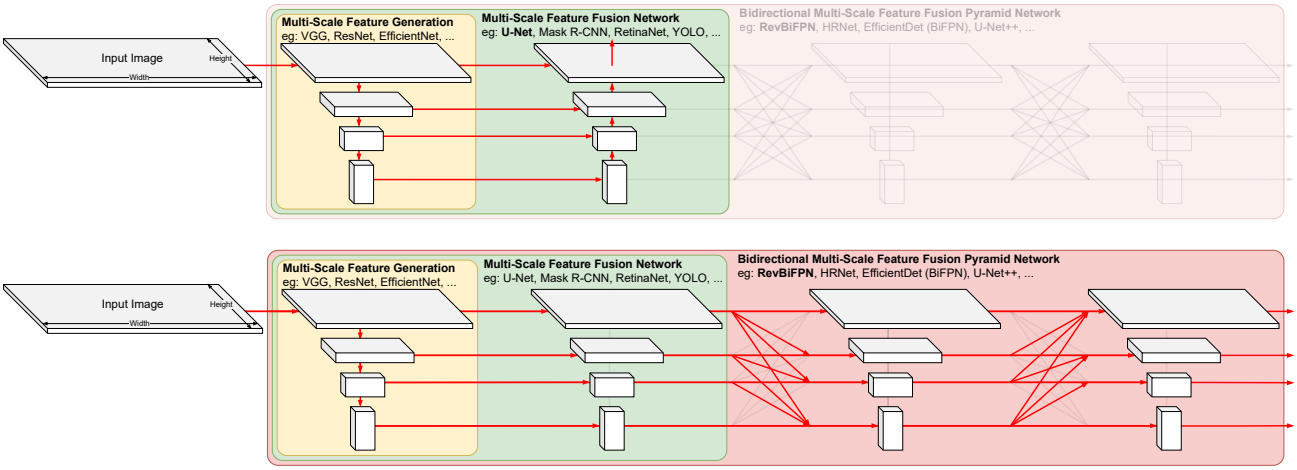

Figure 15. Effective multi-scale connectivity of U-Net (Top) and RevBiFPN (Bottom).

A Examples Multi-Scale Connectivity

Figure 2 encapsulates the connectivity of many network types. Neural Networks have directed connectivity where the network instantiation dictates the direction of the flow of information and is therefore ambiguous unless pertaining to a particular network instance. With an instantiation, the connectivity can be directed. Figure 15 show two examples of instantiated connectivity highlighted in red. The connectivity in the red box can be iteratively applied to further bridge the semantic gap between consecutive feature blocks. For example, EfficientDet (Tan et al., 2020) can be formed by adding all-to-all connectivity across each feature scale and block.

B Reversible Stacked Hourglass Networks

The reversible residual block (Gomez et al., 2017) has only been applied to networks with constant hidden dimensionality. Stacked Hourglass (Newell et al., 2016) networks are built using a stack of hourglass structures that maintain constant dimensionality. Placing each hourglass structure inside a reversible residual block allows the network to produce high-resolution feature maps without storing hidden activations. To enable comparisons with RevBiFPN variants, we implement a fully reversible Stacked Hourglass Network, RevSHNet. RevSHNet uses the MBConv block, a SpaceToDepth stem, channel counts similar to RevBiFPN-S0 channel counts, and a comparable classification head.

B.1 Memory

Even with reversible recomputation enabled, RevSHNet needs to store an entire hourglass of activations. An input size of 224 results in a memory usage increase of about 40% when compared to RevBiFPN (Figure 16).

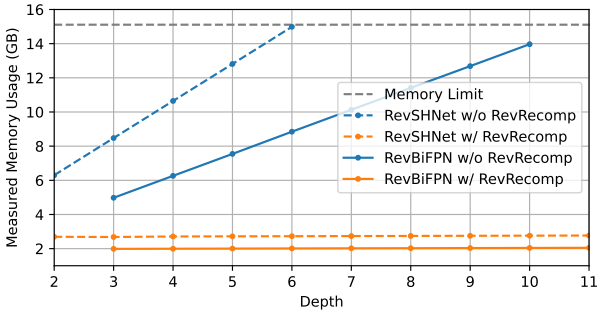

Figure 16. Memory used by RevBiFPN and RevSHNet as depth is scaled with and without reversible recomputation (RevRecomp).

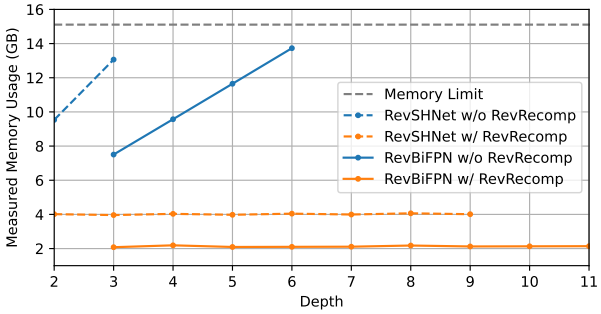

Figure 17. Recreates Figure 16 with an input resolution of 288. RevBiFPN becomes more favorable as resolution is scaled.

B.2 Compute Complexity

When the input size is increased to 288, RevSHNet uses almost twice the memory used by RevBiFPN (Figure 17). The increased memory usage limits memory savings and how much the network can be scaled.

When RevSHNet is scaled, the produced network has a high compute complexity (Figure 18). This is potentially not optimal and wasteful when scaling networks to larger

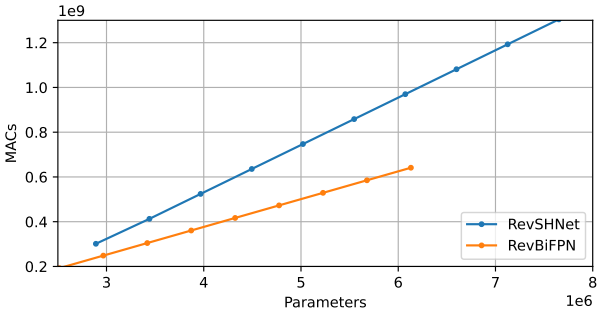

Figure 18. MACs vs Parameter count of RevBiFPN and RevSHNet as depth is scaled.

sizes. It should be noted that the above analysis does not consider network performance. Given comparable networks, we expect RevBiFPN to outperform RevSHNet since RevBiFPN has full bidirectional multi-scale feature fusion with a feature pyramid output, whereas RevSHNet does not.

C REVSILO WITH ADDITIVE COUPLING

While g_j can be any invertible coupling function, as shown in Figure 19 this work uses additive coupling and $F_j = \sum_i F_{i,j}(x_i)$.

For instance Equation (4) is computed as:

$$h_7 = h_3 + (F_{2,7}(h_2) + F_{1,7}(h_1) + F_{0,7}(h_0)) \quad (17)$$

where the equivalent inverse equation computes

$$h_3 = h_7 - (F_{2,7}(h_2) + F_{1,7}(h_1) + F_{0,7}(h_0)). \quad (18)$$

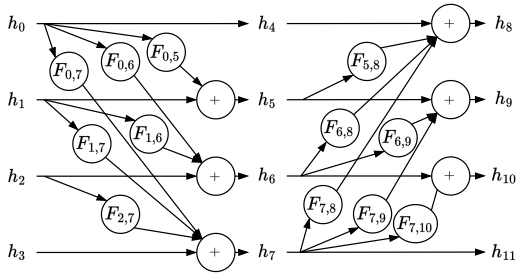

Figure 19. A RevSilo using additive coupling.

D CITYSCAPES SEMANTIC SEGMENTATION

We present experimental results on the Cityscapes semantic segmentation dataset (Cordts et al., 2016). The dataset contains 5000 high-quality pixel-level finely annotated images. The finely annotated images are divided into 2,975/500/1,525 for training, validation, and testing. There are 30 classes, and 19 among them are used for evaluation. The mean of class-wise intersection over union (mIoU) is adopted as the evaluation metric.

Setup. We follow the same training protocol as Zhao et al. (2017; 2018). The data are augmented by random cropping (from 1024 x 2048 to 512 x 1024), random scaling in the range [0.5, 2], and random horizontal flipping. We use the SGD optimizer with a momentum of 0.9 and the weight decay of $5 * 10^{-4}$. RevBiFPN-S0 through S3 use a base learning rate of 0.05; the rest of the networks use a base learning rate of 0.02. The poly learning rate policy with a power of 0.9 is used with a minimum learning rate of 10^{-4}. All the models are trained for 90k iterations with a batch size of 16 across 8 GPUs and using SyncBN. The models are trained using the MMSegmentation open-source segmentation toolbox (MMSegmentation Contributors, 2020). We use the provided FCN and OCR heads to evaluate the capability of the backbone for segmentation tasks.

Results. Table 10 shows the results of training RevBiFPN on the Cityscapes dataset in terms of parameters, computational complexity (evaluation MACs per sample[5]), memory used during training, and mIoU. Although competitive, RevBiFPN can require larger model sizes to be as performant as the HRNet baseline. Instead of using open-source training configurations and segmentation heads, future work can look at improving RevBiFPN's performance (per resource used) for semantic segmentation.

Table 9. Initial dropout and RandAugment ops applied (N). Training initially uses a weight decay of 4×10^{-5}, label smoothing = 0.1, a RandAugment magnitude of 9, and mstd is set to 0.5, and the network is trained without mixup, CutMix, or stochastic depth.

MODEL	DROPOUT	N
REVBIFPN-S0	0.25	2
REVBIFPN-S1	0.25	2
REVBIFPN-S2	0.25	2
REVBIFPN-S3	0.25	2
REVBIFPN-S4	0.4	4
REVBIFPN-S5	0.4	4
REVBIFPN-S6	0.5	5

E IMAGENET REGULARIZATION

Training is regularized using label smoothing (Szegedy et al., 2016), weight decay, dropout (Srivastava et al., 2014), stochastic depth (Huang et al., 2016), CutMix (Yun et al., 2019), mixup (Zhang et al., 2018), and the timm library (Wightman, 2019) variant of RandAugment (Cubuk et al., 2020). Regularization increases with the network scale to prevent larger scales of the network from overfitting. Without knowing how much augmentation was needed for each network, training began with the regularization in Table 9.

Table 10. Semantic segmentation results on the Cityscapes `val` set.

Backbone	FCN Head				ORC Head			
	Params	MACs	Mem	mIoU	Params	MACs	Mem	mIoU
RevBiFPN-S0	1.88M	52.72B	0.65	72.8	5.66M	503.5B	2.71	73.9
RevBiFPN-S1	2.97M	85.41B	0.74	74.1	7.49M	632.6B	2.95	75.7
RevBiFPN-S2	7.00M	200.78B	1.60	75.6	13.35M	990.5B	3.46	76.9
RevBiFPN-S3	14.36M	354.64B	2.01	77.4	22.34M	1356.9B	4.18	79.0
RevBiFPN-S4	39.52M	843.35B	3.10	79.3	51.04M	2309.5B	5.47	80.4
RevBiFPN-S5	68.04M	1463.74B	5.05	80.3	82.88M	3364.7B	6.21	80.8
RevBiFPN-S6	122.56M	2364.37B	6.17	80.4	140.86M	4719.6B	6.44	81.8
HRNetV2-W40 (Wang et al., 2020)	45.89M	536.46B	3.31	80.2	-	-	-	-
HRNetV2-W48 (Wang et al., 2020)	65.86M	748.68B	3.33	81.1	70.3M	1206.3B	4.84	81.6

Table 11. Weight decay (WD), dropout, number of RandAugment ops applied (N), mixup, CutMix, and stochastic depth used at the end of training. Label smoothing uses a coefficient of 0.1 and RandAugment uses a magnitude of 9 and mstd of 0.5.

Model	WD	DropOut	N	Mixup	CutMix	Stochastic Depth
RevBiFPN-S0	4×10^{-5}	0.25	2	0.00	0.0	0.00
RevBiFPN-S1	4×10^{-5}	0.25	2	0.00	0.0	0.00
RevBiFPN-S2	4×10^{-5}	0.3	3	0.00	0.0	0.00
RevBiFPN-S3	4×10^{-5}	0.3	3	0.10	1.0	0.05
RevBiFPN-S4	2×10^{-5}	0.4	4	0.10	1.0	0.10
RevBiFPN-S5	2×10^{-5}	0.4	4	0.20	1.0	0.10
RevBiFPN-S6	2×10^{-5}	0.6	5	0.20	1.0	0.30

When the validation accuracy of the EMA model began to plateau, the regularization of the models was increased. The final regularization used for each network is shown in Table 11.

F ImageNet Model Comparisons

Table 12 extends Table 3. This enable comparisons of RevBiFPN variants with other state of the art networks trained on ImageNet-1K.

G MS COCO Extended Results

Tables 13 and 14 provide the numerical details of Figures 9 to 14 extending the results of Section 5.2.

Table 12. Models trained using only ImageNet-1K. While most networks are trained using 300 to 400 epochs, HRNet and RegNetY use a 100 epoch training schedule.

Model	Params	Train Res	Res	MACS	Top1
RevBiFPN-S0	3.42M	224	224	0.31B	72.8%
RevBiFPN-S1	5.11M	256	256	0.62B	75.9%
RevBiFPN-S2	10.6M	256	256	1.37B	79.0%
RevBiFPN-S3	19.6M	288	288	3.33B	81.1%
RevBiFPN-S4	48.7M	320	320	10.6B	83.0%
RevBiFPN-S5	82.0M	352	352	21.8B	83.7%
RevBiFPN-S6	142.3M	352	352	38.1B	84.2%
EfficientNet-B0 (Tan & Le, 2019)	5.3M	224	224	0.39B	77.1%
EfficientNet-B1 (Tan & Le, 2019)	7.8M	240	240	0.70B	79.1%
EfficientNet-B2 (Tan & Le, 2019)	9.2M	260	260	1.0B	80.1%
EfficientNet-B3 (Tan & Le, 2019)	12M	300	300	1.8B	81.6%
EfficientNet-B4 (Tan & Le, 2019)	19M	380	380	4.2B	82.9%
EfficientNet-B5 (Tan & Le, 2019)	30M	456	456	9.9B	83.6%
EfficientNet-B6 (Tan & Le, 2019)	43M	528	528	19B	84.0%
EfficientNet-B7 (Tan & Le, 2019)	66M	600	600	37B	84.3%
EfficientNet-B5 (Cubuk et al., 2020)	30M	456	456	9.9B	83.9%
EfficientNet-B7 (Cubuk et al., 2020)	66M	600	600	37B	85.0%
EfficientNetV2-S (Tan & Le, 2021)	24M	128 - 300	300	8.8B	83.9%
EfficientNetV2-M (Tan & Le, 2021)	55M	128 - 380	380	24B	85.1%
EfficientNetV2-L (Tan & Le, 2021)	121M	128 - 380	380	53B	85.7%
NFNet-F0 (Brock et al., 2021)	72.0M	192	256	12B	83.6%
NFNet-F1 (Brock et al., 2021)	133M	224	320	36B	84.7%
NFNet-F2 (Brock et al., 2021)	194M	256	352	63B	85.1%
NFNet-F3 (Brock et al., 2021)	255M	320	416	115B	85.7%
NFNet-F4 (Brock et al., 2021)	316M	384	512	215B	85.9%
NFNet-F5 (Brock et al., 2021)	377M	416	544	290B	86.0%
VOLO-D1 (Yuan et al., 2021)	27M	224	384	22.8B	85.2%
VOLO-D2 (Yuan et al., 2021)	59M	224	384	46.1B	86.0%
VOLO-D3 (Yuan et al., 2021)	86M	224	448	67.9B	86.3%
VOLO-D4 (Yuan et al., 2021)	193M	224	448	197B	86.8%
VOLO-D5 (Yuan et al., 2021)	269M	224	448	304B	87.0%
VOLO-D5 (Yuan et al., 2021)	269M	224	512	412B	87.1%
ViT-B/16 (Dosovitskiy et al., 2021)	86.0M	384	384	55.4B	77.91%
ViT-L/16 (Dosovitskiy et al., 2021)	307M	384	384	191B	76.53%
Swin-T (Liu et al., 2021)	29M	224	224	4.5B	81.3%
Swin-S (Liu et al., 2021)	50M	224	224	8.7B	83.0%
Swin-B (Liu et al., 2021)	88M	224	384	47.0B	84.5%
CoAtNet-0 (Dai et al., 2021)	25M	224	384	13.4B	83.9%
CoAtNet-1 (Dai et al., 2021)	42M	224	384	27.4B	85.1%
CoAtNet-2 (Dai et al., 2021)	75M	224	384	49.8B	85.7%
CoAtNet-2 (Dai et al., 2021)	75M	224	512	96.7B	85.9%
CoAtNet-3 (Dai et al., 2021)	168M	224	384	107B	85.8%
CoAtNet-3 (Dai et al., 2021)	168M	224	512	203B	86.0%
CaiT-XXS-24 (Touvron et al., 2021)	12.0M	224	384	9.5B	80.4%
CaiT-XXS-36 (Touvron et al., 2021)	17.3M	224	384	14.2B	81.8%
CaiT-XS-24 (Touvron et al., 2021)	26.6M	224	384	19.3B	83.8%
CaiT-XS-36 (Touvron et al., 2021)	38.6M	224	384	28.8B	84.3%
CaiT-S-24 (Touvron et al., 2021)	46.9M	224	384	32.2B	84.3%
CaiT-S-36 (Touvron et al., 2021)	68.2M	224	384	48.0B	85.0%
CaiT-S-48 (Touvron et al., 2021)	89.5M	224	384	63.8B	85.1%
CaiT-M-24 (Touvron et al., 2021)	185.9M	224	384	116.1B	84.5%
CaiT-M-36 (Touvron et al., 2021)	270.9M	224	384	173.3B	84.9%
HRNet-W18-C (Sun et al., 2019a)	21.3M	224	224	3.99B	76.8%
HRNet-W30-C (Sun et al., 2019a)	37.7M	224	224	7.55B	78.2%
HRNet-W32-C (Sun et al., 2019a)	41.2M	224	224	8.31B	78.5%
HRNet-W40-C (Sun et al., 2019a)	57.6M	224	224	11.8B	78.9%
HRNet-W44-C (Sun et al., 2019a)	67.1M	224	224	13.9B	78.9%
HRNet-W48-C (Sun et al., 2019a)	77.5M	224	224	16.1B	79.3%
HRNet-W64-C (Sun et al., 2019a)	128M	224	224	26.9B	79.5%
RegNetY-200MF (Radosavovic et al., 2020)	3.2M	224	224	0.2B	70.4%
RegNetY-400MF (Radosavovic et al., 2020)	4.3M	224	224	0.4B	74.1%
RegNetY-600MF (Radosavovic et al., 2020)	6.1M	224	224	0.6B	75.5%
RegNetY-800MF (Radosavovic et al., 2020)	6.3M	224	224	0.8B	76.3%
RegNetY-1.6GF (Radosavovic et al., 2020)	11.2M	224	224	1.6B	78.0%
RegNetY-3.2GF (Radosavovic et al., 2020)	19.4M	224	224	3.2B	79.0%
RegNetY-4.0GF (Radosavovic et al., 2020)	20.6M	224	224	4.0B	79.4%
RegNetY-6.4GF (Radosavovic et al., 2020)	30.6M	224	224	6.4B	79.9%
RegNetY-8.0GF (Radosavovic et al., 2020)	39.2M	224	224	8.0B	79.9%
RegNetY-12GF (Radosavovic et al., 2020)	51.8M	224	224	12.1B	80.3%
RegNetY-16GF (Radosavovic et al., 2020)	83.6M	224	224	15.9B	80.4%
RegNetY-32GF (Radosavovic et al., 2020)	145.0M	224	224	32.3B	81.0%

Table 13. Object detection results on COCO `minival` in the Faster R-CNN framework. LS = learning schedule. 1x = 12 epochs, 2x = 24 epochs. Mem = GPU memory used during training. RevBiFPN performs better than HRNet and ResNet on small (AP_S), medium (AP_M), and large (AP_L) objects while using fewer MACs and less training-time memory. Evaluation MACs per sample are reported using the training image resolution (800×1333).

Backbone	Params	MACs	Mem	LS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RevBiFPN-S0	19.55M	135.12B	1.67 GB	1x	31.4	51.5	33.3	17.8	34.3	40.9
RevBiFPN-S1	20.48M	140.66B	1.78 GB	1x	32.0	52.0	34.1	18.3	35.7	43.0
RevBiFPN-S2	23.86M	157.42B	2.13 GB	1x	36.3	57.4	39.3	20.8	39.6	46.6
RevBiFPN-S3	30.40M	180.99B	2.61 GB	1x	38.7	60.0	41.4	23.1	42.0	50.4
RevBiFPN-S4	52.88M	251.02B	4.05 GB	1x	40.3	60.5	44.0	23.7	44.3	52.4
RevBiFPN-S5	77.83M	328.91B	5.50 GB	1x	41.3	62.7	44.8	24.8	45.6	52.5
RevBiFPN-S6	127.51M	465.43B	7.37 GB	1x	42.2	63.5	45.8	25.7	46.5	54.0
HRNetV2p-W18 (Wang et al., 2020)	27.48M	196.18B	6.25 GB	1x	36.2	57.3	39.3	20.7	39.0	46.8
HRNetV2p-W18 (Wang et al., 2020)	27.48M	196.18B	6.25 GB	2x	38.0	58.9	41.5	22.6	40.8	49.6
HRNetV2p-W32 (Wang et al., 2020)	47.28M	298.96B	8.62 GB	1x	39.6	61.0	43.3	23.7	42.5	50.5
HRNetV2p-W32 (Wang et al., 2020)	47.28M	298.96B	8.62 GB	2x	40.9	61.8	44.8	24.4	43.7	53.3
HRNetV2p-W48 (Wang et al., 2020)	83.36M	481.92B	11.64 GB	1x	41.3	62.8	45.1	25.1	44.5	52.9
HRNetV2p-W48 (Wang et al., 2020)	83.36M	481.92B	11.64 GB	2x	41.8	62.8	45.9	25.0	44.7	54.6
ResNet-50-FPN (Wang et al., 2020)	41.53M	216.70B	3.61 GB	1x	36.7	58.3	39.9	20.9	39.8	47.9
ResNet-50-FPN (Wang et al., 2020)	41.53M	216.70B	3.61 GB	2x	37.6	58.7	41.3	21.4	40.8	49.7
ResNet-101-FPN (Wang et al., 2020)	60.52M	296.58B	5.43 GB	1x	39.2	61.1	43.0	22.3	42.9	50.9
ResNet-101-FPN (Wang et al., 2020)	60.52M	296.58B	5.43 GB	2x	39.8	61.4	43.4	22.9	43.6	52.4

Table 14. Instance segmentation and detection results on COCO `minival` in the Mask R-CNN framework. LS = learning schedule. 1x = 12 epochs, 2x = 24 epochs. Mem = GPU memory used during training. RevBiFPN outperforms HRNet for detection and segmentation small (AP_S) and medium (AP_M) objects, as well as bounding box AP for medium (AP_M) and large (AP_L) objects, while using fewer MACs and less training-time memory. Evaluation MACs per sample are reported using the training image resolution (800×1333).

Backbone	Params	MACs	Mem	LS	Mask				BBox			
					AP	AP_S	AP_M	AP_L	AP	AP_S	AP_M	AP_L
RevBiFPN-S0	22.2M	188.2B	2.0GB	1x	29.7	13.5	32.3	44.2	31.4	17.8	34.3	40.9
RevBiFPN-S1	23.1M	193.7B	2.4GB	1x	31.0	14.1	33.3	45.3	34.0	19.0	37.0	44.6
RevBiFPN-S2	26.5M	210.5B	2.6GB	1x	33.7	16.0	35.9	49.2	37.1	21.7	40.2	48.5
RevBiFPN-S3	33.0M	232.9B	2.6GB	1x	35.5	17.4	38.4	50.9	39.4	23.6	43.1	50.9
RevBiFPN-S4	55.5M	304.1B	4.1GB	1x	37.1	17.8	40.1	53.4	41.5	24.2	45.4	53.9
RevBiFPN-S5	80.5M	382.0B	5.5GB	1x	37.8	18.5	40.7	54.3	42.2	25.5	46.3	54.3
RevBiFPN-S6	130.2M	518.5B	7.4GB	1x	38.7	19.8	41.7	55.2	43.3	26.9	47.4	55.6
HRNetV2p-W18 (Wang et al., 2020)	30.1M	249.3B	6.7GB	1x	33.8	15.6	35.6	49.8	37.1	21.9	39.5	47.9
HRNetV2p-W18 (Wang et al., 2020)	30.1M	249.3B	6.7GB	2x	35.3	16.9	37.5	51.8	39.2	23.7	41.7	51.0
HRNetV2p-W32 (Wang et al., 2020)	49.9M	352.0B	9.0GB	1x	36.7	17.3	39.0	53.0	40.9	24.5	43.9	52.2
HRNetV2p-W32 (Wang et al., 2020)	49.9M	352.0B	9.0GB	2x	37.6	17.8	40.0	55.0	42.3	25.0	45.4	54.9
ResNet-50-FPN (Wang et al., 2020)	44.2M	269.8B	4.2GB	1x	34.2	15.7	36.8	50.2	37.8	22.1	40.9	49.3
ResNet-50-FPN (Wang et al., 2020)	44.2M	269.8B	4.2GB	2x	35.0	16.0	37.5	52.0	38.6	21.7	41.6	50.9
ResNet-101-FPN (Wang et al., 2020)	63.2M	349.7B	5.8GB	1x	36.1	16.2	39.0	53.0	40.0	22.6	43.4	52.3
ResNet-101-FPN (Wang et al., 2020)	63.2M	349.7B	5.8GB	2x	36.7	17.0	39.5	54.8	41.0	23.4	44.4	53.9